ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Výuková aplikace Dráček II – Vývoj jádra aplikace pro žákovskou část
Student:	Patrik Pavelec
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Dráček II je dotyková vzdělávací aplikace pro OS Android pro žáky prvního stupně základní školy, mířena především na děti s poruchami učení. Aplikace navazuje na desktopovou aplikaci Dráček, která byla předmětem bakalářských prací z minulých let.

1) Analyzujte funkcionalitu dostupných řešení jiných výukových aplikací.
2) Aktualizujte požadavky učitelů ZŠ Smečno na uživatelské možnosti aplikace (použití OS Android).
3) Analyzujte způsoby implementace modularity v systému Android.
4) Navrhněte:
* způsob komunikace mezi moduly cvičení a jádrem,
* implementační model jádra,
* uživatelské rozhraní žákovské části aplikace,
* řešení offline funkcionality aplikace.
5) Implementujte jádro dle návrhu.
6) Hotové řešení podrobte vhodným testům (jednotkové a integrační) a vyhodnoťte výsledky testování.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 6. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Výuková aplikace Dráček II – Vývoj jádra aplikace pro žákovskou část

Patrik Pavelec

Vedoucí práce: Ing. Jiří Chludil

6. února 2017

Poděkování

Rád bych poděkoval vedoucímu této práce Ing. Jiřímu Chludilovi, který tento projekt vedl po dobu téměř dvou let a přinesl mnoho návrhů na vylepšení Dráčka, které byly později převedeny do praxe. Dále bych rád poděkoval každému členovi mého týmu za spolupráci, která probíhala po celou dobu v dobrém duchu. V neposlední řadě bych své poděkování chtěl věnovat členům nového týmu Dráčka, kteří mi pomohli s úpravou modulu, který byl následně použit na testování. Kromě zmíněných kolegů bych rád poděkoval všem svým přátelům za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. února 2017 .....

Odkaz na tuto práci

Pavelec, Patrik. *Výuková aplikace Dráček II – Vývoj jádra aplikace pro žákovskou část*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací jádra výukové aplikace, kterou budou využívat žáci základní školy, trpící poruchami vnímání. Aplikace bude primárně určena pro tablety se systémem Android.

Jádro slouží k přístupu k jednotlivým cvičením, zobrazení výsledků a zadaných úkolů. Celá aplikace je navržena modulární architekturou, díky čemuž se část práce zabývá různými formami komunikace mezi jednotlivými komponenty. Dále se práce zaměřuje na komunikaci se serverem a zpracování přijímaných dat. Poslední část řeší problém offline spuštění, způsobů autentizace a následné automatické synchronizace dat se serverem.

Při řešení byla zvolena možnost modularity pomocí oddělených APK souborů, mezi kterými se komunikuje pomocí android Intentů. Pro umožnění offline funkcionality jádro ukládá dočasné JSON soubory do privátního aplikačního prostoru zabezpečeného přímo zvolenou platformou Android.

Jádro se povedlo naimplementovat s veškerými hlavními požadovanými funkcionalitami, zadání práce bylo tedy splněno. Na aplikaci se nyní mohou těšit žáci základní školy Smečno, kteří ji budou moci využívat každý den v mnoha různých hodinách.

Klíčová slova mobilní výuková aplikace, základní škola, jádro, mezimodulární komunikace, komunikace se serverem, tablet, android

Abstract

This bachelor thesis contains an analysis and implementation of core functionality of an educational application that will be used by students on a primary school for pupils with perception disorders. The application will be targeted for tablets with operation system Android.

The core will provide access to separate exercise modules, it will display achieved results and assigned tasks by teachers. The whole application is designed using modular architecture, because of which a part of this work is dedicated to exploring different forms of communication between the used components. Next part is about communication with server, accepting data and its proper utilization. The last part is aimed to resolving the problem of offline run, means of offline authentication and automatized synchronization of data with the server.

The modularity was solved by using separate APKs which can communicate using android Intents. The offline run uses saving of temporary JSON files into private application storage space which is secured directly by the android platform.

The core was implemented successfully with all the desired functionalities, the specification was fulfilled in full extend. The students of the primary school Smečno can look forward to using it every day in many different classes.

Keywords mobile educational application, primary school, core, module communication, server communication, tablet, android

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Používání technologických pomůcek při výuce na školách se stává čím dál více populární. Výjimkou není ani ZŠ Smečno, kde studují především děti s poruchou vnímání. Pro zpestření a zefektivnění výuky proto vznikl nápad na výukovou aplikaci, která bude obsahovat mnoho druhů cvičení, zaměřených na různé psychologické poruchy. Z tohoto nápadu vznikl v rámci FIT ČVUT projekt s pracovním názvem Dráček.

S tímto projektem jsem se poprvé setkal díky předmětu Softwarové Inženýrství a přišel mi natolik zajímavý, že jsem se rozhodl v něm pokračovat i v rámci bakalářské práce. Dráček již existoval v první verzi, určené na počítač. Na ZŠ Smečno se však nově objevila možnost výuky na tabletech se systémem Android, která je pro děti daleko zábavnější a flexibilnější. Začal jsem se tedy zabývat možnostmi, jakými by se dala aplikace vylepšit, zefektivnit a především převést na Android platformu. Nová aplikace ponese název Dráček II.

Aplikace bude navržena modulární architekturou, veškeré dílčí cvičení budou tedy fungovat jako zcela samostatné celky. Konkrétně se tato práce zaměřuje na návrh a možnosti jádra, které bude zajišťovat autentizaci, veškerou komunikaci mezi komponenty a výměnu výsledků a dalších dat se serverem. Aplikace bude rozdělena na dvě oddělené části, první pro studenty a druhá pro učitele. Tato práce se nezabývá návrhem učitelské části.

Historie projektu

Projekt Dráček začal v roce 2012 v rámci předmětu Softwarový projekt I, jehož hlavním cílem je využití dosavadních znalostí a dovedností k vytvoření funkčního programu o větším rozsahu, než je běžné pro semestrální práce. Tento projekt vyžaduje aplikaci metod softwarové analýzy a efektivní komunikace, dělby práce a celkové spolupráce v týmu.

Již původní dráček byl od počátku vytvářen ve spolupráci se základní školou Smečno, kde chtěli umožnit žákům atraktivnější způsob řešení úloh

v hodinách pomocí výukové aplikace, původně cílené na počítače. Název Dráček vznikl z místní legendy o drakovi, hlídajícím poklad, za který měl být postaven smečenský zámek. Projekt od počátku vykazoval velký potenciál, v případě úspěchu se nabízely možnosti jeho rozšiřování o další cvičení pro žáky a navazování spolupráce s dalšími školami.

První funkční verze vznikla v roce 2013 v rámci bakalářských prací členů prvního týmu ze Softwarového projektu. Aplikace byla vytvořena v Javě pomocí modulární architektury, kde bylo odděleno jádro od modulů cvičení. Již v této verzi byly k dispozici také jejich editory pro úpravy a přidávání[1].

Zde uvádím členy původního vývojářského týmu a jejich zaměření v rámci projektu:

- **Jan Bradáč** [1] – Tvorba jádra, menu a správa přihlášených uživatelů uvnitř aplikace.
- **Petr Kubišta** [2] – Správa zásuvných modulů žákovské aplikace.
- **Ondřej Kužela** [3] – Persistence dat a serverová aplikace, ukládání výsledků cvičení a informací o uživatelích.
- **Michael Bláha** [4] – Tvorba cvičení, primárně zaměřených na práci s textem a jeho vnímáním.
- **Radek Tomšů** [5] – Tvorba cvičení, primárně zaměřených na zrakové vjemy.

Aplikace byla úspěšně nasazena a používána. Problémem se však ukázal být nedostatek počítačů dostupných žákům, Dráčka mohlo v jednu chvíli používat jen několik z nich. V roce 2014 se však na školu díky evropskému grantu dostaly tablety se systémem Android v dostatečném počtu, aby měl při výuce každý žák svůj vlastní. Vznikl tedy požadavek na implementaci dráčka na tablety, který se začal realizovat opět v rámci Softwarového projektu 1 a 2. Tento nový projekt byl pojmenován Dráček II a pracovali na něm, spolu se mnou, další 3 kolegové. Opět platí, že všichni účastníci pokračovali v projektu i v rámci svých bakalářských prací, aby ho dovedli ke zdárnému konci.

Nový tým byl rozdělen takto:

- **Patrik Pavelec** - Jádro aplikace, stará se o přihlášení uživatele, spouštění modulů a komunikační provázání ostatních částí aplikace.
- **Ondřej Filip** [6] - Serverová část aplikace, poskytující persistenci dat. Modul rozhraní pro učitele.
- **Miroslav Mazel** [7] - Sada modulů včetně editorů 2
- **Michal Bureš** [8] - Sada modulů včetně editorů 1

Ještě před koncem vývoje první verze Dráčka II se přidali další mladší studenti, kteří se rozhodli navazovat a vytvořit další moduly a zpestřit tak možnosti a využití pro výuku.

Cíl práce

Cílem literární rešerše této práce je prozkoumání současné situace v oboru výukových aplikaci, sběr obecných požadavků na software tohoto druhu a analýza technologických řešení, které je pro vývoj na Android platformu možné použít.

Cílem praktické části je vytvoření funkčního jádra aplikace Dráček II, které bude schopné autentizace uživatele, také bude umět spouštět cvičení, kterým bude schopné zaslat zadání a naopak od nich získat výsledky, dále komunikovat se serverem a vhodným způsobem studentovi zpřístupnit jeho dosavadní výsledky.

Slovník pojmů

Pro komunikaci v rámci týmu bylo třeba zavést určité pojmy s jasným významem, které se mohou v mé práci vyskytnout. Následují nejdůležitější z nich, spolu s jejich významem.

- **Původní aplikace Dráček** - Jedná se o první verzi aplikace Dráček vyvinutou v roce 2013 prvním týmem zabývajícím se tímto projektem. Aplikace běžela na platformě Java a byla určena na stolní počítače.
- **Modul cvičení nebo jen cvičení** - Jeden modul zaměřený na určitou problematiku, který uživatel může spustit a vypracovat. Příkladem muže být modul Diktát.
- **Zadání** - Konkrétní instance reálného zadání, která bude studentovi předložena k vypracování v rámci konkrétního modulu. Pro modul Diktát, muže být zadání Doplňování y/i nebo Doplňování e/je.
- **Jádro** - Část rozhraní určena žákům, kde bude možné spouštět jednotlivé moduly, zobrazovat výsledky a kontrolovat své úkoly.
- **Rozhraní pro učitele** - Část rozhraní poskytující učitelům možnost editovat zadání, vypisovat výsledky studentů či zadávání úkolů.
- **Server** - Vzdálená část aplikace, se kterou komunikuje jádro a rozhraní pro učitele, sloužící k persistenci dat.

Kapitola 1

Analýza

1.1 Rozbor zadání

V této části bych rád v krátkosti rozebral jednotlivé body zadání a popsal, jakým způsobem budou v práci jednotlivě vyřešené či adresované.

1. **Analyzujte funkcionalitu dostupných řešení jiných výukových aplikací.**

   Před počátkem projektu je vždy důležité zjistit, jaké řešení problému, jež má tento projekt umožňovat jsou nyní k dispozici a zda se do projektu vůbec pouštět. V rámci tohoto kroku projdu několik momentálně dostupných výukových aplikací na platformu Android a zjistím, jaké možnosti nabízejí žákům, jaké učitelům a zhodnotím ji na základě předem vybraných kritérií.

2. **Aktualizujte požadavky učitelů ZŠ Smečno na uživatelské možnosti aplikace (použití OS Android).**

   V tomto kroku projdu požadavky na funkcionalitu, které jsem obdržel od učitelů základní školy Smečno, zhodnotím jejich splnitelnost a zhotovím z nich jednotlivé diagramy užití, které budou po implementaci testovány.

3. **Navrhněte**

   a) **Způsob komunikace mezi moduly cvičení a jádrem.**
   Modul jako nezávislá jednotka musí být schopná přijmout od jádra informace o tom, jaké zadání spustit. V mnoha z nich jsou také potřeba obrázky, se kterými student musí něco udělat, například setřídit, vybrat nehodící se a podobně. Tyto obrázky se také nesmí nacházet přímo v instalačním balíčku, aby je učitel mohl upravovat a přidávat bez potřeby aktualizace modulu, jsou tedy přijímány od jádra ze serveru, stejně jako zadání. V této části se budu věnovat

celkově možnostem implementace modularity a následné možné komunikaci.

b) **Implementační model jádra.**

Bude vytvořen implementační model dle doporučených postupů softwarového inženýrství.

c) **Uživatelské rozhraní žákovské části aplikace.**

Bude třeba navrhnout uživatelské rozhraní a ovládací prvky aplikace tak, aby nepůsobila složitě a dokázali se v ní vyznat i ti nejmenší žáci, zároveň však tak, aby byla možná veškerá vyžadovaná funkcionalita. V této části budu prezentovat navržené wireframy, podle kterých bude grafická stránka Dráčka II implementována.

d) **Řešení offline funkcionality aplikace.**

Jedním z požadavků na aplikaci, které budou podrobně rozepsány v následující části této práce, bylo, aby šel Dráček spustit i bez přístupu na internet. I v dnešní době se může stát, že dítě doma bezdrátový internet nemá a nebylo by si tak schopné vyplnit zadané úkoly, či prezentovat své úspěchy rodičům. Vzhledem k tomu, že celý návrh byl již od původního dráčka koncipován jako klient-server architektura, která na přístupu k internetu závisí, nebyl tento požadavek zcela jednoduchý na splnění. Zde budou tedy popsány problémy, které kvůli offline módu vznikly a jejich zvolená a odůvodněná řešení.

4. **Implementujte jádro dle návrhu.**

   Jádro bude implementováno na platformu Android. Implementace bude k této práci přiložena na externím médiu.

5. **Hotové řešení podrobte vhodným testům (jednotkové a integrační) a vyhodnoťte výsledky testování.**

   Aplikaci bude třeba řádně otestovat. V případech, kdy to bude dávat smysl, budou vytvořeny jednotkové testy. Ty se týkají částí, které provádějí výpočty či jiné interní procesy s jasně definovanými vstupy a očekávanými výstupy.

   Dále proběhnou takzvané integrační testy, ověřující, zda je jádro schopné komunikace se serverem a moduly podle domluvených protokolů.

   Jako poslední přijdou na řadu testy akceptační, budou ručně otestovány všechny případy užití, které vyplynou z funkčních a nefunkčních požadavků z části analýzy. Toto testování bude probíhat přímo na zařízení Samsung Nexus 10, na které bude jednou aplikace naostro nasazena. Pokud všechny případy užití bez problémů proběhnou, bude aplikace považována za dokončenou.

1.2 Rozbor funkčních a nefunkčních požadavků

Funkční požadavky žákovské části aplikace

- **FP1 - Aplikace bude schopná autentizace uživatele**
  - Autentizace bude probíhat zadáním jména a hesla.
  - Aplikace nebude umožňovat vytvoření nového profilu, žákovské profily budou vytvořeny učiteli v jejich části aplikace.
- **FP2 - Po přihlášení bude možné zapnout výukové moduly**
  - Zobrazeny budou veškeré moduly, které jsou žákovi přiřazeny učitelem a všechny k nim patřící zadání
- **FP3 - Po dokončení modulu se uloží výsledek**
  - Po dokončení modulu se výsledek uloží vždy, nikoliv pouze v případě dokončení zadaného úkolu.
  - Pokud nebude mít aplikace v době dokončení modulu přístup k internetu, uloží se výsledek po opětovném připojení. Výsledek tedy nebude ztracen. (viz. **NP5**)
- **FP4 - Výsledky bude možné zobrazit**
  - Dráček bude obsahovat speciální obrazovku s výsledky, rozdělenými dle modulů se zobrazenými záznamy o dosažených výsledcích. Ty budou zobrazeny procentuálně.
- **FP5 - Aplikace bude umožňovat zobrazení zadaných úkolů s možností jejich spuštění**
  - Další speciální obrazovka bude zobrazovat zadané úkoly rozdělené dle vyučujícího.
  - U každého záznamu bude tlačítko, které okamžitě vybraný úkol spustí.

Nefunkční požadavky žákovské části aplikace

- **NP1 - Aplikace bude cílena na platformu Android verze 6.0**
  - Běh aplikace bude zaručen pro zařízení s verzí androidu 6.0 (Marshmallow) a vyšší. Jedná se tedy o SDK API verzi 23+.
- **NP2 - Ukládání a čtení veškerých dat bude primárně ze serveru**
  - Pokud to bude možné, aplikace bude komunikovat se vzdáleným serverem, ze kterého bude číst vše potřebné pro autentizaci uživatele, stahovat moduly a cvičení a na který bude ukládat výsledky.

- **NP3 - Uživatelské rozhraní bude navrženo pro děti základní školy**
  - Grafická část aplikace bude určena pro malé děti. Měla by tedy být jednoduchá, intuitivní a působit zábavně, ale ne příliš odtrhávat pozornost od cvičení.
- **NP4 - Instalace nových modulů bude probíhat automaticky**
  - Nové moduly budou instalovány automaticky a to s co nejmenším možným zapojením žáků.
  - Jako přijatelné se bere nutnost potvrzení formuláře s instalací, nikoliv však složitá navigace přes Google Play a podobně.
- **NP5 - Aplikace bude spustitelná také v offline režimu**
  - Offline režim bude poskytovat stejnou funkcionalitu co se týče spuštění cvičení a úkolů, výsledky budou odeslány na server po opětovném připojení k internetu.
- **NP6 - Grafické rozhraní dráčka bude primárně určeno pro landscape zobrazení**
  - Aplikace bude umožňovat automatické otáčení, avšak při návrhu rozmístění jednotlivých ovládacích prvků se bude počítat s tím, že bude ve většině času používána v režimu landscape.

1.3 Analýza dostupných řešení

Jak bylo již zmíněno, v této části práce se budu věnovat sběru a rozboru informací o nyní dostupných výukových aplikacích. Zvláštní důraz bude kladen na to, jakou celkovou funkcionalitu umožňují žákům, zda dokáží ukládat výsledky a zpřístupnit je učitelům, jak probíhá autentizace žáka a jakým způsobem probíhají aktualizace aplikace. Na základě této analýzy a nasbíraných funkčních a nefunkčních požadavků bude určeno, zda má tento projekt smysl a zda by nešel o nějaké další vhodné funkcionality obohatit.

1.3.1 Omezení výběru:

Vzhledem k nepřebernému množství dostupných výukových aplikací není možné analyzovat všechna existující řešení. Je tedy třeba vhodným způsobem omezit, jakými aplikacemi se bude konkrétně analýza v této práci zabývat. Vybraná omezení by měla zajistit zkoumání pouze relevantních řešení, která splňují všechna základní kritéria, které jsou na Dráčka II kladeny. Tyto požadavky vyplynuly jak z požadavků na původního Dráčka tak na Dráčka II a dále z požadavků Jiřího Chludila, jakožto vedoucího celého projektu.

Zajímat nás tedy nadále budou pouze aplikace splňující následující podmínky:

- Jsou dostupné z Android marketu k datu 26.12.2016
- Nacházejí se v sekci vzdělávání
- Jsou v českém jazyce
- Jsou zdarma, nebo alespoň dostupné se zkušební licencí

Z takovýchto aplikací budou vybrány 4 zastupující, které budou vyzkoušeny a podrobeny zkoumání nabízené funkcionality. Dále bude následovat rozbor původního dráčka.

1.3.2 Hodnocená kritéria

U vybraných aplikací nás bude zajímat především následující:

- **Možnost autentizace uživatele**

  Jedná se o klíčovou funkcionalitu, která je nezbytná k reálnému využití aplikace ve školách nejen k procvičování, ale i pro skutečné hodnocení studentů. Pokud aplikace nenabízí možnost jednoznačného přihlášení se jako konkrétní student, nelze spárovat jednotlivé výsledky a osobu, která jich dosáhla.

- **Ukládání výsledků**

  Stejně jako první bod, ukládání výsledků je naprosto nezbytný požadavek, bez kterého aplikace může sloužit jen pro zpestření výuky a procvičení. Tato funkcionalita je nezbytná pro učitele ke známkování, pro žáka, aby věděl, kde má své silné stránky a naopak největší mezery a v neposlední řadě také pro rodiče, kteří mají možnost sledovat pokrok svých dětí.

- **Vzdálený přístup k výsledkům**

  Vzdálený přístup k výsledkům není naprostou nutností, ale velice zjednoduší práci učitele v hodinách, když nebude muset studenty jednoho po druhém procházet při dokončení nějakého cvičení. Místo toho si výsledky jedním kliknutím společně stáhne do svého tabletu.

- **Přenositelnost aplikace**

  Vzhledem k tomu, že tablety užívané při reálné výuce jsou téměř vždy vlastnictvím školy, nikoliv žáka, je třeba, aby aplikace nebyla závislá na přístroji, na kterém momentálně běží. Není totiž možné zajistit, že žák bude výhradním uživatelem jednoho zařízení. Všechny informace, které aplikace využívá, by tedy měly být tedy uložené vzdáleně a přistupovat by k nim měla na základě autentizace uživatele.

- **Intuitivní uživatelské rozhraní**

  Mnoho aplikací je určeno pro velice mladé žáky, je tedy podstatné, aby se v nich i ti dokázali bez problému orientovat a aby správně pochopili, co se po nich v jakém cvičení chce.

- **Možnost úpravy cvičení na míru**

  Je jednoduché si představit, že aby byla výuková aplikace opravdu v hodinách použitelná, učitelé by měli mít možnost přizpůsobit nabízená cvičení tomu, co zrovna probírají. Aplikace by jim měla umožnit cvičení přidávat, odebírat, upravovat či prohazovat jejich pořadí.

1.3.3 Vybrané aplikace

Duolingo: Naučte se anglicky [10]

První vybraná aplikace odpovídající našemu omezení se jmenuje Duolingo a specializuje se na výuku jazyků. S více než 3,7 miliony stažení se jedná o jednu z nejúspěšnějších momentálně dostupných výukových aplikací vůbec. Dá se stáhnout v mnoha různých jazycích včetně češtiny, avšak pro každý jazyk nejsou dostupné všechny kurzy. Nejvíce jich je pro anglicky mluvící žáky. Pro česky mluvící je však momentálně dostupná pouze výuka angličtiny.

Výuka je rozdělena do mnoha na sebe navazujících lekcí (viz obr. 1.1). Pokročilou lekci nemá uživatel možnost spustit, dokud úspěšně nedokončí všechny předcházející. Za správná řešení dostává uživatel virtuální měnu, která slouží jako ukazatel pokroku jednotlivce[11]. Lekce obvykle zabere kolem 15 minut a skládá se z několika fází. Jako první je učení nových slovíček pomocí obrázků, následuje překlad vět z/do učeného jazyka za pomoci sady nabídnutých slovíček (viz obr. 1.2) a jako poslední je překlad věty v cizím jazyce do vlastního bez pomoci.

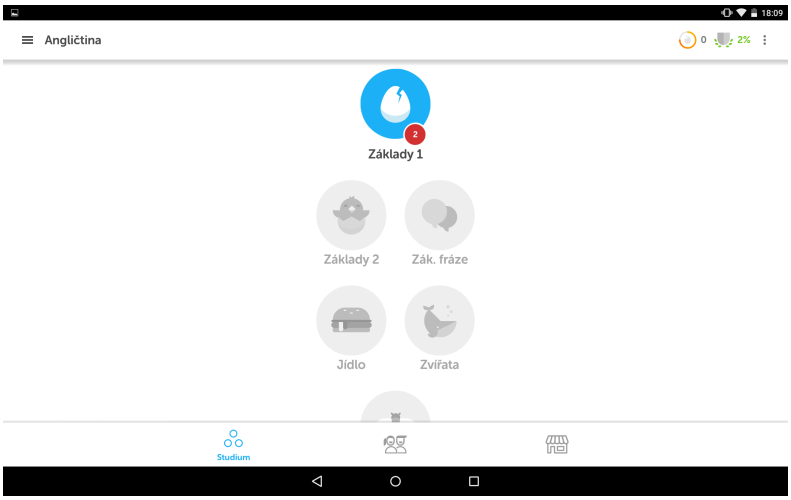
Kladné hodnotím celkovou přehlednost aplikace, která je dokonce doplněna mluvenými instrukcemi. Duolingo však neumožňuje přihlášení, je tedy zcela závislé na přístroji, na kterém je nainstalované. Co se týče zobrazení výsledků, jediné co je zobrazeno jsou odemčené lekce a počet získaných mincí. Jednotlivé lekce jsou zcela neupravitelné.

Výukové kartičky [12]

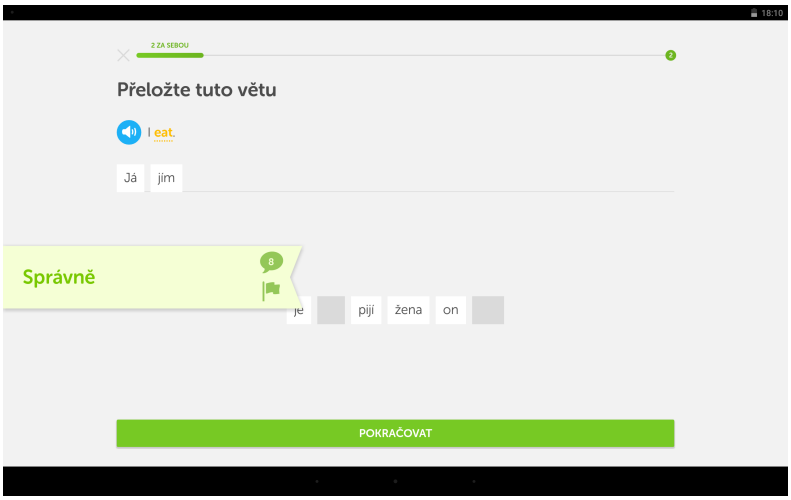
Druhá vybraná aplikace je založena na rozpoznávání věcí na obrázkových kartičkách. Po spuštění si může žák vybrat druh obrázků, které bude rozpoznávat. Na výběr jsou například barvy, divoká zvířata, ovoce, povolání a podobně. V neplacené verzi je k dispozici pouze 6 těchto kategorií, v placené jich je však přes 30.

Po zvolení kategorie se začnou zobrazovat slova, která jsou navíc nahlas přečtena. Žák pak musí identifikovat správnou kartičku z šesti možností, na

Obrázek 1.1: Duolingo - strom lekcí



Obrázek 1.2: Duolingo - překlad



Obrázek 1.3: Výukové kartičky - výsledky

které je napsaná věc zobrazena. K dispozici je možnost opětovného přečtení hledané věci a dále nápovědy, při jejímž využití je správná kartička označena. Pokud však uživatel nápovědu využije či dojde k chybné odpovědi, kolo se nepočítá jako úspěšné. K dokončení jednoho cvičení je třeba úspěšně identifikovat 14 kartiček.

Tato aplikace má dle mého názoru výborně a hravě vyřešené uživatelské rozhraní se spoustou grafických prvků, které však pouze zvyšují celkovou přehlednost a atraktivitu. Dále je k dispozici detailní přehled výsledků, kde jsou zaznamenány nejen správně či chybně vybrané kartičky, ale také čas, který uběhl, než k volbě došlo (viz obr. 1.3).

Bohužel ani tato aplikace neumožňuje přihlášení a výsledky jsou ukládány pouze lokálně. Kartičky není možné přidávat a v neplacené verzi jich je k dispozici jen velice omezené množství.

Počítání pro děti [13]

Tato aplikace je určena pro žáky základních škol, kteří se začínají učit počítat v desítkové soustavě. Je možné si vybrat z celkových 8 cvičení, které jsou určené k procvičování číslic. Celá hra je provázena pouze mluvenými instrukcemi, nepočítá se tedy s tím, že děti budou již umět číst.

Jedním z nabízených cvičení je například vybarvování určitého počtu obrázků. Na začátku se na obrazovce objeví psaná podoba čísla a dítě musí kliknout na stejný počet prozatím šedivých obrysů náhodného obrázku pod ním. Po kliknutí se obrázek vybarví. Problémem v tomto cvičení se zdá být, že počet nevybarvených počátečních obrázků je vždy stejný, jako počet vyžadovaných na obarvení. Žák tedy nemusí nad cvičením vůbec přemýšlet, vždy stačí kliknout postupně na všechny obrysy, které se na začátku objeví (viz obr. 1.4). Tento

Obrázek 1.4: Počítání pro děti - vybarvování

princip, kdy dítě nemusí vyvinout žádnou snahu, se objevuje ještě v několika dalších cvičeních.

Povedenější jsou pak cvičení, ve kterém dítě vybírá menší hodnotu ze dvou zobrazených, opět pomocí obrázků, nebo cvičení, ve kterém musí vybrat obrázek s konkrétním počtem předmětů, určeným číslicí v horní části obrazovky.

Co se týče nabízené funkcionality, je naprosto minimální. Aplikace neumožňuje přihlášení či jiné rozlišení uživatele, neukládá výsledky ani lokálně a nelze ji žádným způsobem rozšířit.

Jdu do školy! [14]

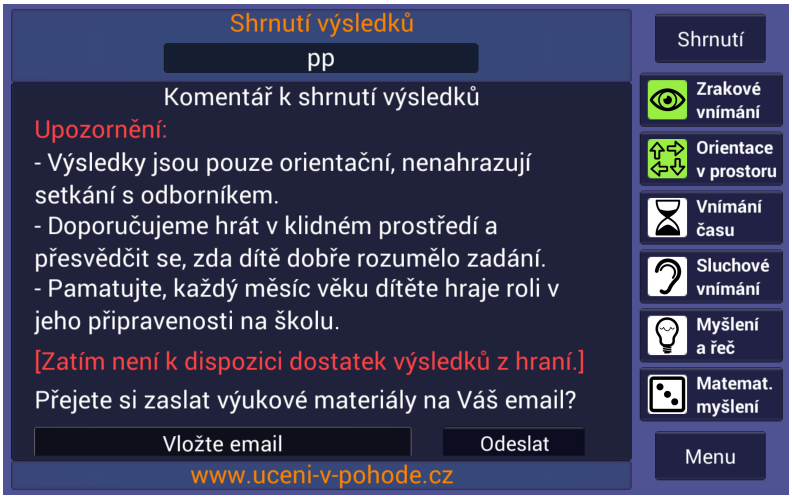
Poslední vybraná aplikace se jmenuje *Jdu do školy!* a vznikla ve spolupráci s FIT, ČVUT. Cílovou skupinou jsou předškoláci kolem 5 let věku, kteří se připravují začít studium v první třídě na základní škole. Tato aplikace zkoumá vyspělost dítěte v několika ohledech, které jsou pro počátek studia důležité a to: zrakové vnímání, orientace v prostoru, vnímání času, sluchové vnímání, myšlení a řeč a matematické myšlení.

Celá hra je koncipována jako cesta po malé planetce s myšákem Adamem (viz obr. 1.6), který je její jediný obyvatel. Myšák pomocí namluvených instrukcí zadává žákovi různé úkoly, které byly speciálně připraveny pro tento účel od psycholožky Mgr. Simony Pekárkové, která se na vývoji celé aplikace podílela [15]. Za každé splněné cvičení žák dostane kapičku do konve, kterou na konci zalije myšákovu slunečnici.

Co se týče nabízených možností, zdají se zde být v celém tomto výběru aplikací nejrozsáhlejší. Na začátku je možné vytvořit si vlastní profil, který jde kdykoliv změnit. Ten slouží k zaznamenávání postupu v příběhu, takže není třeba ho dohrát během jednoho sezení. Konkrétní výsledky jednotlivých

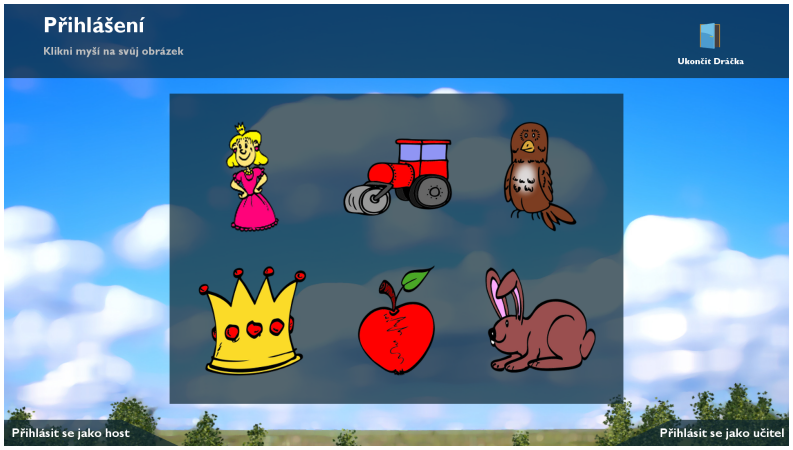
Obrázek 1.5: Jdu do školy - menu

Obrázek 1.6: Jdu do školy - výsledky



disciplín nelze v menu zobrazit číselně, místo toho je však zobrazen stupeň vyspělosti žáka pomocí 4 barevně odlišených kategorií, kde zelená znamená, že žák je v tomto ohledu zcela připraven (viz obr. 1.6). Kategorie postupují přes žlutou a oranžovou k červené, která značí, že má dítě v tomto ohledu ještě velké mezery. Co se týče rozšiřitelnosti, tak ta ani zde není zcela k dispozici, avšak v menu lze vypnout a zapnout různé možnosti jako například předvádění správného řešení před začátkem hry či zobrazení řešení při chybě.

Obrázek 1.7: Dráček - přihlášení



Původní Dráček

Původní Dráček jako jediný neodpovídá výběrovým kritériím, jelikož není určen na platformu Android. Protože však na něj tato bakalářská práce přímo navazuje, je nezbytné, aby byla provedena analýza funkcionalit, které obsahoval, abychom novou verzí o žádné důležité nepřišli.

Již od počátku byl projekt brán jako komplexnější výukový systém s mnoha různými a na sobě nezávislými cvičeními. Nebylo o něm však uvažováno jako o nástroji, který bude sloužit učitelům ke klasifikaci studentů, pouze jako o nástroji k zábavnějšímu a názornějšímu procvičování látky. Proto ani v první implementaci není možnost autentizace uživatele. Existuje však šest různým uživatelských profilů rozlišených podle obrázku, který je na počátku aplikace zvolen.(viz obr. 1.7) Profily nelze přidávat ani odebírat. Po zvolení profilu se žák dostane do hlavního menu, kde může pokračovat na výběr se cvičeními, úspěšné a neúspěšné výsledky nebo na povídání o legendě o smečenském drakovi. Celé uživatelské rozhraní je poskládáno z mnoha různých obrázků a působí přehledně. Menu je zobrazeno jako křižovatka, výběr cvičení pak představují okna v budově školy (viz obr. 1.8).

Již tato verze je založena na klient-server architektuře, je tedy přenositelná a výsledky jsou ukládány vzdáleně, nikoliv lokálně. Ke každému cvičení byl připraven jednoduchý editor pro učitele, který umožňoval jejich úpravu.

1.3.4 Vyhodnocení

Z informací k jednotlivým vybraným aplikacím je na první pohled jasné, že jsou určené pouze jako pomocný materiál při výuce. Jsou tedy jen doplňkem, který je vhodný k vizualizaci či usnadnění pochopení probírané látky. Žádná z těchto aplikací však není konstruována tak, jak bychom si přáli, aby jednou Dráček II vypadal, tedy jako komplexní a všeobecně využitelný výukový systém. I tak se

Obrázek 1.8: Dráček - výběr cvičení

však jedná o povedené a užitečné pomůcky při výuce. Jednou z hlavních věcí, která momentálně v dráčkovi chybí, je podpora hlasových instrukcí, které velice zjednodušují pochopení aplikace pro nejmenší studenty. Nad touto možností bude třeba se v budoucnu zamyslet. Především se však týká jednotlivých cvičení a nikoliv jádra, na které je tato práce zaměřena.

Tabulka 1.1 zobrazuje mé hodnocení jednotlivých aplikací dle hodnotících kritérií. Ke každému z nich jsem přidělil hodnotu 0 až 2, kde 0 značí naprosto nesplněno a 2 splněno bez výtek.

Jedním z cílů projektu Dráček II je, splňoval všechna tato kritéria co nejlépe. Věřím, že tak vznikne aplikace umožňující procvičování různorodé látky bez specifického zaměření, která bude mít potenciál stát se přínosnou v naprosté většině předmětů a zaujmout proto místo v každodenní výuce. S funkcionalitami zjednodušujícími klasifikaci pro učitele a možnostmi úprav a přidávání cvičení se navíc stane daleko nezávislejší na vývojářích, než doposud existující aplikace.

1.4 Případy užití

Diagram případu užití, zobrazený na obrázku 1.9, slouží k vizualizaci a lepšímu pochopení funkčních požadavků. Detailněji zobrazuje chování aplikace na různé uživatelské podněty. Vzhledem k tomu, že kreslit speciální diagram užití pro každý funkční požadavek by vedlo k vytvoření mnoha malých a zbytečně jednoduchých instancí, rozhodl jsem se, že veškerou hlavní funkcionalitu shrnou do jediného obsáhlejšího diagramu. Věřím, že tento způsob bude přehlednější.

Tabulka 1.1: Vyhodnocení dostupných řešení

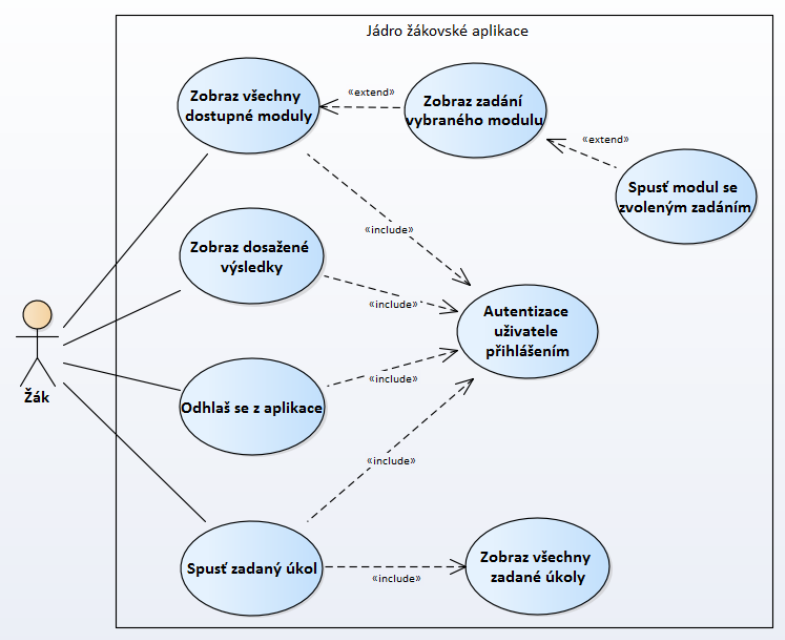
	Možnost autentizace uživatele	Ukládání výsledků	Vzdálený přístup k výsledkům	Přenositelnost aplikace	Intuitivní uživatelské rozhraní	Možnost úpravy cvičení na míru	Celkem
Duolingo	0	1	0	0	2	0	**3/12**
Výukové kartičky	0	2	0	0	2	0	**4/12**
Počítání pro děti	0	0	0	0	2	0	**2/12**
Jdu do školy!	0	1	0	0	2	1	**4/12**
Původní Dráček	0	1	1	2	2	2	**8/12**

1.4.1 Definice persón

V našem diagramu bude vystupovat pouze jediná persóna, kterou je **žák**. Ta představuje studenta základní školy, který bude Dráčka využívat v hodinách pro procvičování, či doma pro plnění domácích úloh. Učitelé budou mít vlastní aplikaci pro zadávání úkolů a spravování žáků, kterou se tato práce nezabývá. Nejsou tedy v diagramu potřeba.

1.4.2 Diagram a popis

Obrázek 1.9: Diagram případů užití



Zapnout určitý modul

Popis:

Žák si spustí vybraný modul na procvičení látky.

Podmínky:

Žák je přihlášen uživatelským jménem a heslem a nachází se v hlavním menu.

Scénář:

1. Žák poklepe na položku v hlavním menu nazvanou Cvičení.

2. Aplikace zobrazí všechny povolené moduly.
3. Žák vybere pomocí obrázku a názvu modul.
4. Aplikace zobrazí všechny dostupná zadání.
5. Žák vybere a poklepe na zadání.
6. Aplikace spustí modul s vybraným zadáním.

Zobrazit výsledky

Popis:

Žák si zobrazí veškeré dosažené výsledky.

Podmínky:

Žák je přihlášen uživatelským jménem a heslem a nachází se v hlavním menu.

Scénář:

1. Žák poklepe na položku v hlavním menu nazvanou Výsledky.
2. Aplikace zobrazí všechny zaznamenané výsledky setříděné podle modulů a dále seřazené dle data.

Odhlásit z aplikace

Popis:

Žák se bezpečně odhlásí z aplikace.

Podmínky:

Žák je přihlášen uživatelským jménem a heslem a nachází se v hlavním menu.

Scénář:

1. Žák poklepe na ikonku vypnutí nacházející se v pravém horním rohu.
2. Aplikace zobrazí potvrzovací dialog s varováním, že po odhlášení bude třeba opětovného zadání jména a hesla a přístupu k internetu.
3. Žák dialog potvrdí potvrzovacím tlačítkem.
4. Aplikace provede ohlášení žáka a ukončí se.

Spustit určitý úkol

Popis:

Žák si zobrazí zadané úkoly od učitelů a vybraný z nich spustí.

Podmínky:

Žák je přihlášen uživatelským jménem a heslem a nachází se v hlavním menu.

Scénář:

1. Žák poklepe na položku v hlavním menu nazvanou Úkoly.
2. Aplikace zobrazí všechny doposud nevyplněné úkoly setříděné dle zadavatele.
3. Žák vybere úkol ke splnění a pomocí tlačítka vedle něho úkol spustí
4. Aplikace spustí modul se zadáním, které jsou specifikované jako úkol.

Kapitola 2

Návrh a implementace

2.1 Modularita

Pro implementaci Dráčka byla zvolena modulární architektura. Každé cvičení je samostatný modul, který vždy komunikuje pouze s jádrem aplikace. Jádro se pak stará o poskytování různých zadání a o ukládání výsledků na server. Tento přístup má mnoho výhod. Tímto způsobem dojde k rozdělení programu na nezávislé, zaměnitelné části, z nichž každá obsahuje vše nezbytné pro svůj vlastní běh. Moduly zajišťují oddělení zodpovědností a zlepšují udržovatelnost kódu.

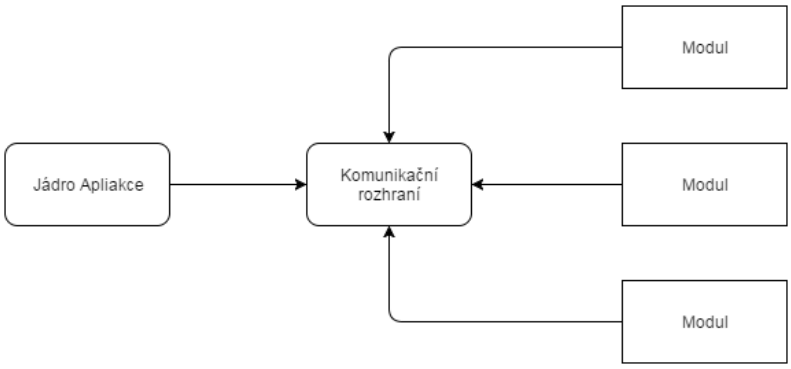
Bylo třeba se rozhodnout, jak k implementaci modularity přistoupit. Jako hlavní možnosti se jevily tyto tři:

1. Stahování prostředků
2. Modularita kontejnerová
3. Oddělené APK

2.1.1 Stahování prostředků

První zvažovanou možností bylo mít veškerý kód v jednom android projektu, ke kterému by měli všichni vývojáři přístup. Jádro by od cvičení bylo odděleno pouze pomocí balíčků. Vytvoření nového modulu v tomto případě by vyžadovalo aktualizaci celé aplikace. Poté, co by se učitelé rozhodli modul použít, by se ze serveru musely automaticky stáhnout veškeré prostředky, jako jsou obrázky, animace a konfigurace, které by modul potřeboval k běhu. Tímto způsobem by aplikace nebyla zbytečně veliká, neboť samotný kód zabírá minimum místa a veliké soubory by se stahovaly až v případě potřeby. Všechen kód by však byl centralizován na jediném místě a musel by se složitě řešit problém udělování oprávnění k zásahům při každém rozšíření projektu. Udržovat a kontrolovat Dráčka tímto způsobem by bylo téměř nemožné. Pracuje na něm zároveň větší množství studentů a to každý na jiné části. Musela by existovat osoba, která by

Obrázek 2.1: Kontejnerová modularita Dráčka



každý zásah do projektu zkontrolovala a schválila, že se jedná pouze o změny v části, za kterou je student zodpovědný, aby nedocházelo k narušení jiné funkcionality, o které dotyčný nemusel vědět. Vzhledem k náročnosti takového úkolu byl tento způsob zavržen.

2.1.2 Modularita kontejnerová

Vzhledem k faktu, že programovací jazyk Android platformy vychází z jazyku Java, nabízí často podobná řešení specifických problémů[16]. V Javě existuje možnost implementace modularity za pomoci nástrojů jako je například Maven či Gradle, podle standardizované specifikace OSGI[17]. Tyto nástroje kromě jiného dokáží vytvářet a spravovat takzvané závislosti projektů. Pomocí řídícího souboru se dá určitá část kódu, zpravidla obsahující Java rozhraní, označit jako viditelná a využitelná navenek pro ostatní projekty, i když se jedná o 2 zcela samostatné .jar soubory. Takové části kódu se říká exportovaná. Ostatní projekty, opět pomocí řídícího souboru Mavenu či Gradlu, deklarují závislost na takovém projektu a mohou volat metody exportovaných rozhraní. Pro běh takovéhoto modulárního projektu je však potřeba takzvaný kontejner, který umožňuje a spravuje samotné volání závislých částí kódu za běhu. Jedním z takových kontejnerů je Felix od organizace Apache[18].

Při zvolení takovéhoto postupu bychom definovali společné rozhraní pro všechny moduly, umožňující jejich spuštění se zvolenou konfigurací a navrácení výsledků, které by bylo poskytnuto zájemcům o jejich implementaci. Jádro Dráčka by mělo definovanou závislost pouze na toto rozhraní a o přístup k implementacím by se staral kontejner. Grafické znázornění závislostí je na obrázku 2.1. Tím bychom se vyhnuli problému nepotřebného a potenciálně nebezpečného sdílení kódu, o kterém jsem mluvil v předchozím řešení. Zmíněný Apache Felix existuje i ve verzi pro Android platformu[19], proto se možnost dosažení modularity tímto způsobem zdála být ze začátku vhodná. Problémem se však ukázala být složitá konfigurace samotného kontejneru při přidávání modulů automaticky, což byl jeden z hlavních požadavků na naši aplikaci. Dále

toto řešení přinášelo potřebu jeho manuální instalace a nastavení v každém zařízení, ve kterém bychom Dráčka chtěli spouštět. Kvůli tomu bylo nakonec také vyloučeno.

2.1.3 Oddělené APK

Posledním zvažovaným řešením bylo vyvíjet moduly jako zcela samostatné Android aplikace (.apk soubory), které by musely dodržovat předem definovaný protokol komunikace. Výhoda tohoto přístupu je, že veřejně známý by musel být pouze tento protokol a mezi jednotlivými moduly by neexistovaly žádné programové závislosti, které by musely být složitě spravované kontejnerem. Tím by bylo umožněno, že by nové moduly mohl také připravit kdokoliv, aniž by musel mít přístup k dosavadnímu kódu jádra či jiných modulů. Vzhledem ke skutečnosti, že komunikace mezi jádrem a moduly není příliš složitá, definovat a kontrolovat takový protokol by nebylo těžké. Stejně jako u předchozího řešení by po přípravě nového modulu nebyla potřeba aktualizace dosavadních, již nainstalovaných aplikací. Veškeré potřebné informace by získaly ze společného serveru, kam by bylo cvičení po dokončení implementace nahráno ve formě instalačního souboru APK.

2.1.4 Zvolené řešení

Jako řešení modularity byla zvolena poslední možnost oddělených APK, která přináší stejné výhody jako modularita kontejnerová a zároveň odstraňuje nutnost uživatelsky nepříjemné správy kontejneru. Pro implementaci tedy bylo třeba zjistit, zda existuje způsob spouštění externích aplikací z jádra a definovat protokol na předávání zadání do modulů a získávání výsledků po dokončení. Kromě tohoto bylo také potřeba ověřit, zda Android nabízí možnost instalace aplikací přímo z modulu jádra.

2.2 Komunikace jádra

2.2.1 Způsob komunikace s moduly

Jak bylo řečeno v předchozí části, kvůli implementaci modularity pomocí oddělených aplikací bylo nezbytné definovat způsob komunikace mezi jednotlivými moduly cvičení a jádrem. Konkrétně se jedná o samotné spuštění modulu, předání zvoleného zadání a potřebných obrázků a přijetí výsledku ke zpracování po ukončení.

Podle Android dokumentace je možné spouštět externí aplikace pomocí tzv. intentu[20]. Intent je objekt, který obsahuje popis činnosti, kterou má samotný operační systém android vykonat. Nejčastěji se používá k zobrazení jiných aktivit (obrazovek) v rámci jedné aplikace, není to však zdaleka jeho jediné využití. Jednou z jeho vlastností je možnost přibalení neomezeného

množství serializovatelných dat pomocí mapy klíčů a hodnot, které jsou při spuštění externí aplikaci k dispozici. Dále umožňuje stejným způsobem vrátit serializovaný výsledek aktivitě, ve které intent vznikl. Takový přenos dat, který zajišťuje a zabezpečuje přímo operační systém, na který bude Dráček určen, je v našem případě naprosto vyhovující a nemělo tedy smysl snažit se vymyslet vlastní řešení. Intent ponese při spouštění modulu zadání jako hodnotu v mapě pod klíčem **EXERCISE_FILE** a pokud bude potřeba k zadání dodat obrázky, budou přístupné v mapě pod klíčem **IMAGES**. Vracet bude modul jediný soubor pod klíčem **RESULTS**.

2.2.2 Formáty přenášených dat

Dalším logickým krokem bylo zamyšlení se nad tím, jak přesně budou vypadat přenášené soubory a zda je potřeba jejich podobu vynucovat. Toto rozhodnutí záleželo na tom, zda bude konkrétní soubor žákovská nebo učitelská část parsovat k použití, nebo jestli ho bude pouze přenášet a ukládat.

Souborem, který bude jádro pouze přenášet je zadání. To je vytvořeno v Editoru, který je součástí samotného modulu a poté předáno učitelské části jádra, která ho uloží na server. Při spuštění je žákovskou částí staženo a předáno modulu, který s ním již nakládá dle svého. V tomto případě tedy není třeba podobu regulovat. Může to tedy být textový soubor s jakýmkoliv obsahem.

Složitější to je s výsledky a obrázky. Výsledky jsou zobrazené jak v žákovské, tak v učitelské části aplikace. Je tedy třeba, aby bylo jádro schopné je po přijetí od modulu správně interpretovat. S obrázky je problém v serializaci, která může být u jakýchkoliv větších souborů velice pomalá. Bylo by tedy vhodné se jí nějakým způsobem vyhnout.

Formát výsledků

Soubor s výsledky bude moci obsahovat daleko větší množství informací, než jen celkovou známku či procenta. Protože je aplikace určena pro děti s poruchami vnímání, je žádoucí, aby moduly mohly měřit i sekundární informace, které mohou sloužit učiteli, či psychologovi k určení problému dítěte. Příkladem může být čas potřebný k dokončení, počet špatných odpovědí, celkový počet kliknutí, prodlevy mezi kliknutími a mnoho dalších.

Bylo tedy jasné, že bude potřeba použít nějaký z komplexnějších strojově zpracovatelných textových formátů, mezi kterými dnes dominuje buď XML nebo JSON. Ukázky obou formátů jsou na obrázku 2.2.

- **XML**

  XML je textový formát využívaný k serializaci dat již desítky let, který je založen na principu pojmenovaných párových tagú. K těm lze přidat kromě hodnoty ještě tzv. atributy, které tag dále specifikují, pokud je to potřeba.[24] Nevýhodou tohoto formátu je, že tagy je třeba zavírat,

Obrázek 2.2: XML vs. JSON formát [22]

což nepřináší žádnou přidanou informační hodnotu, ale nelze bez nich soubor jednoznačně naparsovat. Velkou výhodou je však existence standardizovaných nástrojů sloužících například k dotazování se na obsah (XPath) a k validaci formátu (Relax NG, Xml Schema) či transformaci dat (XSL Transformation).

- **JSON**

  JSON je strojově zpracovatelný textový formát, který vznikl z XML kolem roku 2000. Je oproštěn o párové tagy, které zvětšují velikost výsledného souboru. Tím však ztrácí možnost přidání atributů a stává se mapou klíčů a hodnot, kde hodnotou může být další vnořené pole.[23] Tento formát se používá především při síťové komunikaci, kde je velikost odesílaných a přijímaných dat kritická pro rychlost aplikace.

Protože formát výsledku musí být zcela jednoznačně daný, byla vybrána reprezentace pomocí XML, ke kterému bude existovat veřejně dostupné XML schema. To bude sloužit jako kontrakt. Pokud budou modulem zasílané výsledky validní podle schematu, bude jádro schopné je přijímat a dále s nimi pracovat. Protože však nelze předem odhadnout, co všechno budou které moduly schopné měřit, bylo schema navrženo tak, aby kontrolovalo povinné elementy, kterými

je celkový výsledek v procentech a celkový čas ve vteřinách a dále povolovalo neomezený počet nepovinných elementů s libovolným obsahem.

Schema výsledků

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="Results">
    <xs:complexType mixed="false">
      <xs:sequence>
        <xs:element name="Time">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="timeFormat">
                <xs:attribute name="name" fixed="Celkový čas" use="required" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      <xs:element name="Score">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="scoreFormat">
              <xs:attribute name="name" fixed="Konečné score" use="required" />
                </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="Specific" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="notEmptyString">
              <xs:attribute name="name" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="timeFormat">
    <xs:restriction base="xs:token">
      <xs:pattern value="[0-9]{1,6}s"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="scoreFormat">
    <xs:restriction base="xs:token">
      <xs:pattern value="[0-9]{1,3}%"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="notEmptyString">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Formát a přenos obrázků

Jak již bylo řečeno, některé moduly vyžadují kromě textového zadání ještě obrázky, se kterými bude žák pracovat. Jako podporované byly vybrány přípony .bmp a .png, což jsou dva oblíbené bezztrátové formáty. Protože se však jedná o větší soubory, které by mohly zabrat delší dobu při serializaci, bylo vhodné vymyslet lepší způsob, jak modulům obrázky zpřístupnit.

První uvažovanou možností bylo po stažení zkopírovat obrázek z aplikačního prostoru na externí uložiště, ke kterému mají přístup všechny aplikace a poslat modulu informaci o jeho umístění. To se však jeví jako neohrabané a opět zbytečně zdlouhavé. Bylo by také potřeba zajistit mazání kopií obrázků, aby nedocházelo k zaplňování externí paměti. Takové řešení by bylo uvažováno až v případě, že nebude existovat lepší možnost.

Naštěstí platforma Android umožňuje povolit přístup externích aplikací k souborům, umístěným v privátní části paměti pomocí tzv. FileProviderů[25]. Ty se deklarují v manifestu projektu a umožňují přesnou specifikaci s jakými soubory je povoleno jakým způsobem manipulovat. K takto zpřístupněným obrázkům je možné vygenerovat URI, což je textová reprezentace jejich umístění a informací o FileProvideru, který je k přístupu k nim potřeba. Modulům bude z jádra zaslán list obsahující URI všech obrázků, které jsou momentálně na serveru k modulu k dispozici. Ve spouštěné aktivitě modulu je pak možné dostat přístup k obrázkům pomocí následujícího kódu:

```
ArrayList<String> urls =
    getIntent().getStringArrayListExtra("IMAGES");
for (String url : urls) {
   Uri uri = Uri.parse(url);
   FileDescriptor r = null;

   try {
      r = getContentResolver()
      .acquireContentProviderClient(uri)
      .openFile(uri,"r")
      .getFileDescriptor();
      } catch (FileNotFoundException e) {
         // Core failed to provide the image, alert user and exit.
       }
       FileInputStream fis = new FileInputStream(r);
       ...
}
```

2.2.3 Komunikace se serverem

Výběr způsobu, jakým bude jádro komunikovat se serverem, nebyl pouze na mě. Bylo třeba ho domluvit s Ondřejem Filipem, který v rámci své bakalářské práce[6] pracoval mimo jiné i na serverové části Dráčka. Ze všech možných

druhů webových služeb jsme vybrali RESTful Web Services (dále jen REST), protože se jedná o v dnešní době nejjednodušší a nejrozšířenější způsob jejich implementace, se kterým jsme již oba měli předchozí zkušenosti a k našim účelům zcela postačoval[26]. Pro více informací o jednotlivých možnostech serverové komunikace čtenáři doporučuji nahlédnout přímo do Ondřejovy práce, jelikož tyto informace nespadají do tématu tohoto textu týkajícího se jádra.

Princip RESTu

REST využívá `http` protokol ke komunikaci mezi aplikací a webovým serverem pomocí klasických požadavků `GET, POST, PUT, DELETE` a dalších. Ty jsou zasílány aplikací na přesně určenou `url` (tzv. `resource`), podle které server pozná, jakou operaci po něm aplikace vyžaduje. Po přijetí požadavku server nahlédne do jeho hlavičky, ve které se mimo jiné nalézá, o jaký druh se konkrétně jedná a informace o tom, jakým způsobem jsou formátovány zaslaná data v těle. Z toho vyplývá, že je možné mít několik funkcionalit přístupných pod jedinou `url`, kde se konkrétní požadovaná akce určí podle toho, jaký druh požadavku na server přijde. V těle mohou, ale nemusí být data, potřebné ke správnému vykonání akce. Požadavky `GET` jsou typicky dotazovací a často žádné informace v těle nenesou. Oproti tomu požadavky `POST` a `PUT` slouží k ukládání nových informací na server, k čemuž v těle musí obsahovat ukládaný obsah. Ten je typicky ve formátu XML, JSON či neformátovaného textu.

REST v Dráčkovi

Jádro aplikace nyní komunikuje se serverem pomocí mnoha druhů požadavků. Každý z nich však nese v hlavičce záznam pojmenovaný `API-API-KEY` s vygenerovaným řetězcem, který je známy pouze serveru a jádru aplikace. Server jakýkoliv požadavek bez tohoto záznamu automaticky zahazuje. Dále každý takový z nich, který vyžaduje autentizaci uživatele, obsahuje záznam s názvem `API-USER-TOKEN` obsahující serverem vygenerovaný řetězec při přihlášení. Tento token má na serveru omezenou dobu platnosti a snižuje riziko odposlechu přihlašovacích údajů, jelikož ty se na server přenášejí pouze jednou za celou dobu přihlášení stejného uživatele. Rozhodli jsme se pro veškerou RESTovou komunikaci mezi serverem a jádrem používat formát JSON, především protože je úspornější než XML a nemá oproti němu v tomto případě zcela žádné nevýhody.

Server pro Dráčka běží na adrese `185.88.73.6`. Následující ukázkový požadavek oznámí serveru, že se do aplikace snaží přihlásit uživatel se jménem `JDoe` a heslem `1234`:

```
--> POST http://185.88.73.6/dracek-nette/www/api/v1/token HTTP/1.1
    Content-Type: application/json; charset=UTF-8
    Content-Length: 36
    API-API-KEY: fc73b664-c712-0470-aa6a-bcb6048a5daf
    {
        "password":"1234",
        "username":"JDoe"
    }
--> END POST (36-byte body)
```

Server ověří tyto informace v databázi, a pokud jméno s heslem odpovídají, vygeneruje token a pošle ho aplikaci spolu s dalšími informacemi o uživateli v následujícím formátu:

```
<-- 200 OK http://185.88.73.6/dracek-nette/www/api/v1/token (386ms)
    Date: Sun, 22 Jan 2017 12:08:12 GMT
    Expires: Mon, 23 Jan 2017 10:00:00 GMT
    Content-Type: application/json;charset=utf-8
    {
        "token": {
            "token": "61bf3fcf99f6038f24aa39f6f7a3f63x",
            "expiration": 1487765292
            },
        "user": {
            "id": 1,
            "birthdate": null,
            "role": "student",
            "name": "John",
            "surname": "Doe",
            "username": "JDoe"
            }
    }
<-- END HTTP (186-byte body)
```

Aplikace odpověď přijme a v případě úspěšného ověření uživatele jej vpustí do zabezpečené části jádra.

2.2.3.1 Implementace RESTu

Pro implementaci RESTových služeb pro platformu Android je k dispozici hned několik knihoven, ze kterých bych zmínil především následující dvě:

- **Volley**

  Jedná se o implementaci RESTové komunikace přímo společnosti Google, která za uživatele zařizuje převod Java objektů `JsonObjectRequest` a

`JsonObjectResponse` do formátu `http` požadavku a sama ho pošle na určitou url. Zasílané entitní třídy však musí uživatel sám konvertovat na `JSONObject` a ručně zde mapovat každý zasílaný atribut.

- **Retrofit**

  Tato open source knihovna má veškerou funkcionalitu oficiálního Volley, ale je uživatelsky daleko příjemnější. Přehledně odděluje deklaraci jednotlivých požadavků pomocí anotací od kódu prováděného při vlastním volání a přijímání odpovědi. Dále také zařizuje plně automatický převod mezi Java objekty a http požadavkem s daty v JSON formátu. Pro implementaci serverové komunikace jsem tedy zvolil tuto možnost.

Při implementaci jsem vytvořil abstraktní třídu zařizující společné vlastnosti všech volání. Určí tedy společnou část url, na kterou bude požadavek poslán a vloží do hlavičky potřebné informace. Podstatná část kódu vypadá následovně:

```
public static String BASE_URL = "http://185.88.73.6/dracek-nette/";
public static String REST_URL = BASE_URL + "www/api/v1/";

public static <S> S~createApi(Class<S> serviceInterface, final String
    token) {
    Retrofit.Builder builder = new Retrofit.Builder()
        .baseUrl(REST_URL)
        .addConverterFactory(GsonConverterFactory.create());

    HttpLoggingInterceptor loggingInterceptor = new
        HttpLoggingInterceptor();
    loggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

    OkHttpClient httpClient = new
        OkHttpClient.Builder().addInterceptor(new Interceptor() {
        @Override
        public Response intercept(Chain chain) throws IOException {
            Request.Builder reqBuilder = chain.request().newBuilder();
            reqBuilder.addHeader("API-API-KEY", "abc");
            if (LoggedUser.getToken() != null &&
                !LoggedUser.getToken().equals("")) {
                reqBuilder.addHeader("API-USER-TOKEN", token);
            }
            return chain.proceed(reqBuilder.build());
        }
    }).addInterceptor(loggingInterceptor).build();

    builder.client(httpClient);
    return builder.build().create(serviceInterface);
}
```

Tato metoda přijme třídu definující rozhraní pro konkrétní volání a vrátí proxy třídu, na které se dají volat její metody. Při volání však zařídí převod vstupních parametrů do JSON formátu a vše nezbytné pro vytvoření http požadavku. Rozhraní pro generování ukázkového požadavku pro získání bezpečnostního tokenu pak vypadá zcela jednoduše:

```
public interface RestLoggingApi {

    @POST("token")
    public Call<LogInResult> logIn(@Body LogInRequest body);

}
```

Java objekt `LogInRequest` obsahuje atributy `username` a `password` s veřejnými gettery a settery a o převod se dále uživatel knihovny nemusí starat. Samotné vyvolání požadavku se vejde na jeden řádek kódu:

```
RestApiGenerator.createApi(RestLoggingApi.class).logIn(request).enqueue(...)
```

Metoda `enqueue` přijímá objekt `Callback`, který definuje akce vykonané po vrácení odpovědi od serveru.

2.3 Instalace modulů

Jedním z požadavků na jádro aplikace bylo, aby dokázalo nějakým způsobem zcela, či alespoň zčásti automatizovat instalaci nových modulů, které budou žáci moci využívat. Bez této funkcionality by museli učitelé ručně na každém zařízení všechny moduly nalézt a nainstalovat, což by bylo uživatelsky velice nepohodlné. Tento požadavek se ukázal být daleko náročnějším, než bylo původně předpokládáno. Pro připomenutí zmíním, že moduly jsou implementovány v podobě samostatné Android aplikace, které jsou instalovány pomocí instalačních balíčků s příponou `.apk`. Takové balíčky jsou typicky ukládány na oficiálním uložišti Google Play. Původní myšlenka byla taková, že jádro bude schopné moduly samostatně stáhnout a nainstalovat zcela bez zásahů koncového uživatele aplikace, který bude pouze informován, že instalace probíhá.

Při implementaci jsem však narazil na překážky způsobené přímo platformou Android. První z nich se ukázala být nemožnost instalace nové aplikace bez explicitního potvrzení uživatele. Jedná se o dialog, ve kterém jsou informace o jejím názvu a o povoleních, které jsou při jejím spuštění vyžadovány. Uživatel má poté možnost buďto instalaci potvrdit, či zamítnout. Prohledal jsem oficiální zdroje, zda existuje možnost, jak se tomuto potvrzení vyhnout, ale nenašel jsem o žádné ani zmínku. Na dvou neoficiálních fórech, zabývajících se řešení programátorských problémů, jsem nalezl informaci o tom, že se jedná

o bezpečnostní opatření, zaručující nemožnost instalace reklamních a škodlivých materiálů na zařízení bez vědomí uživatele, které není možné obejít. Ani tuto informaci jsem však na oficiálních zdrojích nenašel k ověření. Vzhledem k tomu, že se mi však nepodařilo najít žádné řešení tohoto problému, jsem byl přinucen tuto skutečnost akceptovat a při instalaci modulů vždy uživatele požádat o potvrzení. Jedná se o jediné kliknutí, které by mělo být pro žáky příliš problematické.

2.3.1 Získání instalačního balíčku

Další nesnází bylo vyřešení způsobu, jakým způsobem instalační balíček na tablet dostat. Původní myšlenka s automatickým stažením z Google Play se ukázala být nesplnitelná z důvodu nedostatečné funkcionality nabízeného oficiálního rozhraní. Pomocí oficiálně podporovaných prostředků je možné vyvolat navigaci do Google Play s přímou navigací na konkrétní aplikaci, není však možné zamezit instalaci jiných aplikací a už zcela nemožné je stažení aplikace bez zásahu uživatele[28]. V případě, že by Dráček byl vydán pod placenou licencí, by se dále objevil problém s nutností přihlášení správným jménem a heslem školy, aby ke stažení mohlo dojít. Některé z těchto problémů dokáže vyřešit neoficiální knihovna s opensource licencí Android Market Api[29]. Použití neoficiálních prostředků k dosažení požadované funkcionality je však vždy riskantní. Nikdo nezaručí, že po jakékoliv aktualizaci Google Play nepřestane tento způsob trvale fungovat a že neobsahuje programátorské chyby, které by mohly ohrozit stabilitu naší aplikace. Při nahlédnutí do git repozitáře je dále vidět, že poslední úprava tohoto api proběhla v roce 2012 a od té doby nebylo udržované. Vzhledem ke všem těmto faktům jsem se rozhodl toto řešení zcela zamítnout, alespoň dokud se neobjeví oficiální cesta, jak námi vyžadované funkcionality dosáhnout.

Alternativním způsobem je ukládat instalační balíčky přímo na našem serveru, se kterým již jádro komunikuje pomocí RESTu. K tomuto způsobu je zapotřebí na každém zařízení manuálně povolit možnost instalace aplikací z neznámých zdrojů, která se nachází v sekci *Nastavení - Zabezpečení - Správa zařízení*. Za neznámý zdroj bere Android jakoukoliv jinou lokaci než Google Play. Toto povolení zvyšuje potenciální riziko nechtěné instalace škodlivého softwaru, avšak jak bylo zmíněno v předchozím článku, toho nelze dosáhnout bez uživatelského potvrzení. Navíc by se již na zařízení musela nějaká taková aplikace nacházet a běžet, aby uživatele k instalaci dalších vybízela. Vzhledem k nedostupnosti jiného řešení a složitým způsobem možnosti zneužití se mi povolení tohoto natavení jeví jako přijatelné.

Prostředky na serveru dedikovaném pro naši aplikaci jsou momentálně nezabezpečené. Vzhledem k tomu, že prozatím není zcela jistá obchodí a distribuční strategie Dráčka, nemohl jsem moduly na veřejný server nahrát. Je možné, že server bude třeba v budoucnu upravit a zabezpečit. Jádro jsem se tedy rozhodl připravit pro stahování modulů za použití nějaké jiné aplikace,

která je momentálně dostupná s volnou licencí. Ukázalo se však, že po půl roce od vývoje serveru již nikdo nezná přístupové heslo a ani po měsíci opakovaných žádostí o jeho restartování mi nebylo vyhověno. Vydal jsem se tedy jedinou zbývající cestou a část kódu, zařizující získání instalačního balíčku jsem byl prozatím nucen nechat neimplementovanou. Kód zařizující jejich instalaci jsem připravil tak, aby bylo tuto funkcionalitu jednoduché doplnit.

2.3.2 Automatická instalace

I přesto, že se získání instalačního `.apk` souboru se ukázalo být komplikovanější, než bylo na první pohled zřejmé, implementace části jádra zařizující automatickou instalaci byla téměř bez problémů. Momentálně se počítá s tím, že instalované balíčky jsou uložené v externí paměti určené pro stažené prostředky, umístěné ve speciální složce s názvem *dragon* a pojmenované pomocí ID modulu ze serveru. Správné pojmenování jednotlivých balíčků je nezbytné, jelikož jádro instaluje moduly až v případě, že jsou učitelem žákovy povolené, nikoliv všechny najednou. Bez způsobu identifikace by nebylo možné správný modul nalézt jinou metodou než pokus, omyl. Je tedy zodpovědností doručovatele balíčků učitelům, aby bylo pojmenování korektní. Manuální zkopírování doručených modulů je záležitostí několika vteřin a je třeba provést pouze jednou při pořízení aplikace a následovně pouze v případě, že se škola rozhodne využívat nějaké nově vyvinuté moduly. Po instalaci dojde ke smazání .apk souboru, jelikož již nebude na daném zařízení potřeba a zbytečně by nezabíral místo v externí paměti. Pro názornost následuje vizuální zobrazení předpokládaného umístění balíčků.

```
Downloads ..... umístění stažených prostředků
├── dragon ..... adresář dedikovaný balíčkům
│   ├── 12.apk ..... instalační balíček pro modul s id 12
│   └── 15.apk ..... instalační balíček pro modul s id 15
└── xxx ..... další nesouvisející stažené soubory
```

Po implementaci automatického získání balíčků bude možné celý tento proces převést do interního souborového prostoru jádra a zvýšit tím bezpečnost instalace. V tuto chvíli je možné instalační balíček ručně zaměnit za jiný, stejně pojmenovaný a jádro ho pak bude považovat za korektní modul. Proběhne tedy jeho instalace, nikdy však neproběhne jeho spuštění, neboť to je dále kontrolováno podle balíčkového identifikátoru, který je taktéž uložen přímo na serveru.

Pro instalaci aplikací pomocí jádra je třeba, aby byly správně nastavené jeho povolení[30]. Ty se definují v manifestu projektu. Pro bezproblémový běh budou potřeba následující:

- `INTERNET`

  Toto povolení umožňuje aplikaci internetovou komunikaci. Bude třeba pro stahování balíčků v budoucnu a veškerou komunikaci se serverem.

- `READ_EXTERNAL_STORAGE`

  Toto povolení umožňuje čtení dat z externí paměti, ve které budou momentálně umístěné instalační balíčky. Po doplnění funkcionality stahování a přesunu balíčků do interní části paměti bude možné toto povolení odstranit.

- `WRITE_EXTERNAL_STORAGE`

  Tímto povolením se umožní smazání balíčků po jejich instalaci. Opět platí, že bude možné ho odstranit stejně jako povolení ke čtení.

- `INSTALL_PACKAGES`

  Poslední povolení je nezbytné k umožnění samotné instalace, která je implicitně platformou Android zakázána z bezpečnostních důvodů.

Pokud jsou všechna tyto povolení správně deklarována, je samotná instalace po nalezení konkrétního balíčku v paměti zcela triviální a v kódu vypadá následovně:

```
Intent promptInstall = new Intent(Intent.ACTION_VIEW)
    .setDataAndType(Uri.fromFile(download),
        "application/vnd.android.package-archive");
context.startActivity(promptInstall);
```

Atribut *download* představuje objekt typu `File` s umístěním balíčku, *context* pak představuje aktivitu, která se o vyvolání instalačního dialogu pokouší.

2.4 Offline mód

V této části se budu zabývat návrhem offline funkcionality Dráčka, která je jedním z požadavků na aplikaci především kvůli žákům, kteří doma nemají přístup k internetu. Pro ně by bez umožnění offline běhu nebylo možné plnit zadané domácí úlohy. Je třeba si však uvědomit, že i když je třeba se těmito případy zabývat, jedná se o nestandardní použití aplikace, které nutně přichází s několika omezeními, aby nebyla narušena její bezpečnost. Kvůli tomuto požadavku je také zakázáno, aby jakýkoliv modul komunikovat přímo přes internet. Veškerá data, která potřebuje ke svému spuštění a dokončení musí nutně získat od jádra.

2.4.1 Autentizace uživatele

Návrh offline běhu vypadá následovně. Aplikace při přihlašování uloží informace potřebné k autorizaci uživatele a bude ukládat provedené změny do lokální databáze. Po připojení zařízení k internetu dojde k automatické synchronizaci a doplnění chybějících dat serveru. Nabízelo se několik možností, jakým způsobem zařídit, aby server věděl, ke kterému uživateli se nově zaslaná data vztahují.

Ukládání hesla

První nejjednodušší možností bylo při přihlášení uložit jméno a heslo na tablet. Takové řešení však představuje bezpečnostní riziko, neboť i přes to, že by informace byly uloženy v privátním aplikačním prostoru, existují metody, jak se k takovému souboru dostat. Nejsou sice nijak jednoduché a vyžadují tzv. *rootování* zařízení, ale je jich možné dosáhnout. Toto tvrzení bohužel nemohu podložit oficiálním zdrojem, avšak existuje několik neoficiálních návodů, jak se k takovému souboru dostat. Další problém pramenil z možnosti, že by někdo mohl omylem zadat jméno či heslo špatně. V takovém případě by se to tablet dozvěděl až po získání přístupu na internet, kdy už by ho mohl používat jiný student. Výsledky by se tedy ztratily. Toto řešení bylo z uvedených důvodů zamítnuto.

Ukládání hash kódu

Jako další možnost se nabízelo ukládat hash otisk místo hesla. Prvním zádrhelem by bylo, že by musela být jednoznačně domluvena hashovací funkce používaná jak na serveru, tak na tabletu, a pokud by byla potřeba její změna, musela by se provést v obou komponentách zároveň.

Dále by bylo potřeba vyřešit, jak by byly takto uložené výsledky zasílány serveru. Nebylo možné udělat podobnou nezabezpečenou službu pro přihlašování pomocí hash otisku místo hesla, neboť by tím byl podkopán celý princip bezpečnosti založený na tom, že i když se potenciální útočník k hash kódům na serveru dostane, nedokáže se s nimi přihlásit. Hashovací funkce fungují pouze jednosměrně a není výpočetně možné z nich heslo odvodit. Bylo by však možné udělat takovou službu zabezpečenou a přístupnou pouze přihlášenému uživateli. To by znamenalo, že by se výsledky synchronizovaly se serverem až v době, kdy by se do Dráčka přihlásil někdo jiný v online režimu.

Tento způsob řešení původní bezpečnostní rizika lehce snížil, nikdo by se nedostal k uloženému heslu, avšak mohl by se dostat k hash kódu. Ten by poté však opět mohl použít pro zaslání výsledků na server, pokud by měl sám přístup do aplikace pod jakýmkoliv jiným uživatelským heslem. Dále také trvá problém s mylným přihlášením. Po této analýze můžeme vidět, že popsané řešení není stále o mnoho lepší, než předchozí.

Token s dlouhou platností

Jako nejschůdnější řešení bylo vybráno používat metodu tokenů, které byly již zmíněny v sekci o komunikaci se serverem, s delší dobou platnosti a rozlišovat jednoduché ukončení aplikace od kompletního odhlášení. Token je generován serverem při žádosti o přihlášení a zaslán zpět aplikaci. Ta ho poté používá v každém RESTovém požadavku, aby server poznal, k jakému studentovi zasílaná data patří.

Samotné přihlášení do aplikace je tedy možné pouze v případě, že je v danou chvíli server dostupný. Pokud by student aplikaci následně ukončil bez odhlášení, zůstal by na serveru i v tabletu token stále platný a i pokud by se od této chvíle zařízení ocitlo bez internetu, bylo by možné ukládat výsledky s příslušným tokenem, který by byl po opětovném připojení použit k jejich zaslání. Pokud by se student odhlásil, přístupový token by byl okamžitě odstraněn z tabletu.

Jedná se o bezkonkurenčně nejbezpečnější řešení z těchto tří. Nevýhoda spočívá v tom, že se bez připojení žák do aplikace nemůže přihlásit, pokud tak již neučinil dříve. Tu je však v tomto případě třeba akceptovat.

2.4.2 Ukládání dat

V této sekci budu řešit kdy, kam a jakým způsobem jsou data do tabletu ukládána.

Při řešení otázky kdy ukládat bylo uvažováno několik možností, mezi kterými bylo i periodické stahování a ukládání všech dat ze serveru na tablet, spuštěné vždy po určitém časovém intervalu. Stahování všech dat najednou však může při pomalé rychlosti internetu trvat až několik vteřin. Bylo by tedy potřeba uživatele vždy vyrušit v práci s tabletem, což se nezdálo být vhodné. Dále zde byla otázka zvolení správného intervalu. Pokud by byl interval příliš dlouhý, mohly by se data při přerušení připojení z tabletu ztratit, i když už je uživatel jednou viděl a mohl s nimi pracovat. Při plánovaném odpojení by tomu mohlo zabránit synchronizační tlačítko, které by bylo umístěno v hlavním menu a na vyžádání by veškeré informace ze serveru stáhlo. Při neočekávaném výpadku by se ztrátě dat nedalo zabránit nijak. I s možností aktualizačního mechanismu jsem se však rozhodl, že opakované vyrušování uživatele je něco, čemu by se aplikace pro děti měla pokud možno vyhnout.

Druhou optimálnější možností je ukládat informace okamžitě po jakémkoliv přijetí serverové odpovědi na jádrem zaslaný požadavek. Vzhledem k tomu, že jediná odpověď obsahuje jen zlomek uložených dat, je jejich příjem a zapsáni na tablet tak rychlé, že ho uživatel ve většině případů ani nepostřehne. Nevýhodou tohoto přístupu je, že se na tablet ukládají pouze informace, na které se uživatel při práci dotazuje. Pokud tedy student za celou vyučovací hodinu nenavštíví obrazovku se zadanými úkoly, nebude ji mít doma bez internetu k dispozici. Tento problém jsem se rozhodl vyřešit již zmíněným aktualizačním tlačítkem,

které zajistí, že pokud ho žák na konci hodiny zmáčkne, bude mít přístup k veškerým informacím dostupným na serveru v daný čas i bez internetu a nemusí se o nic dalšího starat. Celková synchronizace dat také proběhne vždy při spuštění aplikace a to včetně odeslání všech výsledků uložených jádrem v době bez připojení. Výsledky se pak dají manuálně zaslat opět v rámci zmiňovaného tlačítka.

Co se týče způsobu uložení dat, byla uvažována možnost lokální databáze. To se však ukázalo být zbytečně robustním řešením, jelikož uchovávat je potřeba jen několik jednoduchých entit. Zvolil jsem tedy možnost objektové serializace do formátu JSON a vytvoření jednoduché struktury složek, kam se budou objekty ukládat. Toto rozložení je graficky zobrazeno v následujícím schématu.

Mapa interní paměti

```
CORE ..... interní aplikační prostor jádra
├── credits ..... adresář pro přihlašovací data
│   └── login_info.json ..... informace o přihlášeném uživateli
└── users_data ..... adresář pro uživatelská data
    ├── 1 ..... adresář dedikovaný uživateli s id 1
    │   ├── exercises ..... adresář pro entity cvičení
    │   │   └── file.JSON ..... konkrétní serializované objekty
    │   ├── modules ..... adresář pro entity modulů
    │   │   └── file.JSON ..... konkrétní serializované objekty
    │   ├── results ..... adresář pro entity výsledků
    │   │   ├── file.JSON ..... konkrétní serializované objekty
    │   │   └── offline ..... adresář pro výsledky čekající k odeslání
    │   │       └── 12013059.JSON .... konkrétní výsledek s časovou značkou
    │   └── tasks ..... adresář pro entity úkolů
    │       └── file.JSON ..... konkrétní serializované objekty
    └── 3 ..... adresář dedikovaný uživateli s id 3
```

2.5 Achievementy

Jedním z hlavních cílů aplikace je, aby působila na děti atraktivně a neodrazovala je od učení. V rámci splnění tohoto požadavku bylo navrženo vylepšení Dráčka oproti původní verzi tím způsobem, že by obsahoval systém získávání tzv. *achievementů* za dosažené úspěchy. Ty by se mohly dále dělit na obecné a specifické pro jednotlivá cvičení.

S tímto požadavkem však vznikla otázka, jak ho vhodně implementačně vyřešit, aby zůstala zachována nezávislost jádra a cvičení. Na první pohled by se zdálo, že vyhodnocování dosažených achievementů by měl zajišťovat každý modul zvlášť a vhodným způsobem výsledek jádru vracet. Pro vyhodnocení některých z nich by však mohla nastat situace, kdy by modul potřeboval znát všechny výsledky a to nejen přihlášeného žáka, ale i všech ostatních. Tato

situace by vznikla například pro vyhodnocení achievementu za nejrychleji splněné zadání.

Uvědomíme-li si však, že komunikace mezi jádrem a moduly probíhá pouze pomocí serializace a deserializace dat a také, že by veškeré výsledky musely být rozeslány všem dostupným modulům při každém otevření obrazovky s achievementy, je jasné, že takové řešení není výkonnostně přijatelné. Tento způsob by také kladl další povinné nároky na vývojáře modulů.

Jako nejlepší řešení bylo vybráno, že tuto funkcionalitu bude obsluhovat zcela samostatný modul aplikace, který bude v případě přidání nového cvičení možné upravit a samostatně aktualizovat na již instalovaných aplikacích. Implementace takového systému odměn však rozsahem přesahuje možnosti této práce a nebude zde tedy dopodrobna řešena. Čtenáře odkáži na bakalářskou práci Karla Kovařovice s názvem *Výuková aplikace Dráček II – gamifikace a personalizace*[9], která se systémem odměn zabývá podrobněji.

2.6 Uživatelské rozhraní

Jako první důležitou informaci k tomuto tématu musím napsat, že jsem v rámci této práce navrhl a vytvořil funkční uživatelské rozhraní se všemi potřebnými komponenty k umožnění veškeré funkcionality, avšak nezabýval jsem se uzpůsobením a zatraktivněním grafiky pro malé děti. Tomu se bude věnovat specializovaný grafik, který je již touto dobou na projekt domluven. Vzhledem k tomu, že jsem se osobně grafikou nikdy nezabýval, nedokázal bych vytvořit grafickou část ani zdaleka tak atraktivní jako profesionál a tato práce by byla zcela zbytečná.

Jedním z důležitých kroků před začátkem samotné implementace je třeba si ujasnit, jak bude vypadat uživatelské rozhraní výsledné aplikace, na což se v softwarovém inženýrství používají tzv. wireframy. Wireframe je diagram komponent, který se vztahuje k jedné konkrétní obrazovce dostupné v aplikaci. Názorně zobrazuje jejich rozložení s poměry velikostí a často vede k ujasnění celé funkcionality. Nezřídka se také stává, že až po sestavení wireframů návrháři dojdou nedostatky aplikace či potenciální problémy, které má možnost ještě před implementací analyzovat a probrat se zákazníkem.

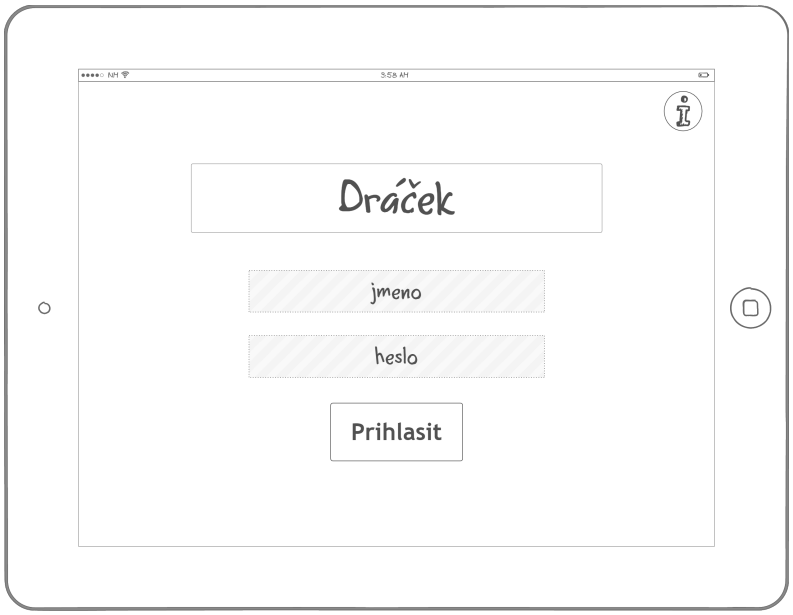
K vytvoření wireframů k jádru Dráčka jsem použil online nástroj `https://ninjamock.com/`, který je zdarma a nabízí velkou škálu použitelných komponent, které se dají zarámovat do několika možných zařízení, včetně tabletu. Potřebné obrázky jsem získal z `https://pixabay.com/` a `http://fontello.com/`, které je nabízejí ke stažení s volnou licencí.

2.6.1 Wireframy

Obrázek 2.3 ukazuje přihlašovací obrazovku aplikace, kde bude student zadávat jméno a heslo. Kromě toho se zde nachází informační tlačítko, které studentovi

poradí kde na jaké obrazovce zrovna je a jaké zde má možnosti. Informační tlačítko je dostupné na každé obrazovce.

Obrázek 2.3: Přihlašovací stránka



Na obrázku 2.4 můžeme vidět hlavní menu, které se objeví po úspěšném přihlášení studenta. V levém horním rohu vidí své jméno, v pravém pak informační tlačítko, možnost odhlášení a kromě viditelného bylo později přidáno aktualizační tlačítko, o kterém jsem mluvil v sekci **Offline mód**. V této části aplikace má uživatel možnost navigace ke spuštění jednotlivých modulů, zobrazení výsledků či zadaných úkolů. Přidal jsem také navigaci na achievementy, které zatím nejsou implementované, ale jednou budou součástí dráčka. Prozatím však tlačítko nezpůsobí žádnou akci.

Obrázek 2.4: Hlavní menu

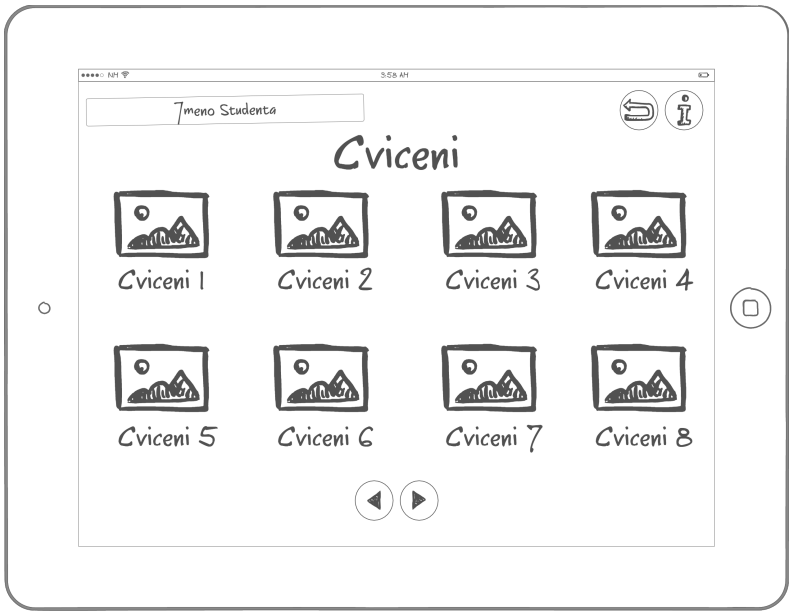
Následující obrázek 2.5 zobrazuje potvrzovací dialog, který se zobrazí po kliknutí na tlačítko ukončení.

Obrázek 2.5: Odhlášení

Pokud student hlavním menu klikne na ikonku *Cvičení*, dostane se na obrazovku představovanou obrázkem 2.6. Zde se nachází veškeré povolené moduly, které jsou zobrazeny buďto pomocí screenshotu uloženém na serveru,

nebo v případě jeho nepřítomnosti implicitním obrázkem. Dále zde přibylo tlačítko na návrat na předchozí obrazovku, které se opět nachází na všech z nich kromě obrazovky menu.

Obrázek 2.6: Povolené moduly



Po rozkliknutí konkrétního modulu se zobrazí obrazovka 2.7, kde jsou vypsané všechny dostupné zadání s popisem, co se v nich po studentovi bude chtít. Ve finální aplikaci bylo vynecháno tlačítko *Spustit*, stačí místo toho poklepat kamkoliv do řádku se zadáním. Následuje spuštění aktivity daného modulu, zatímco jádro bude čekat na výsledek.

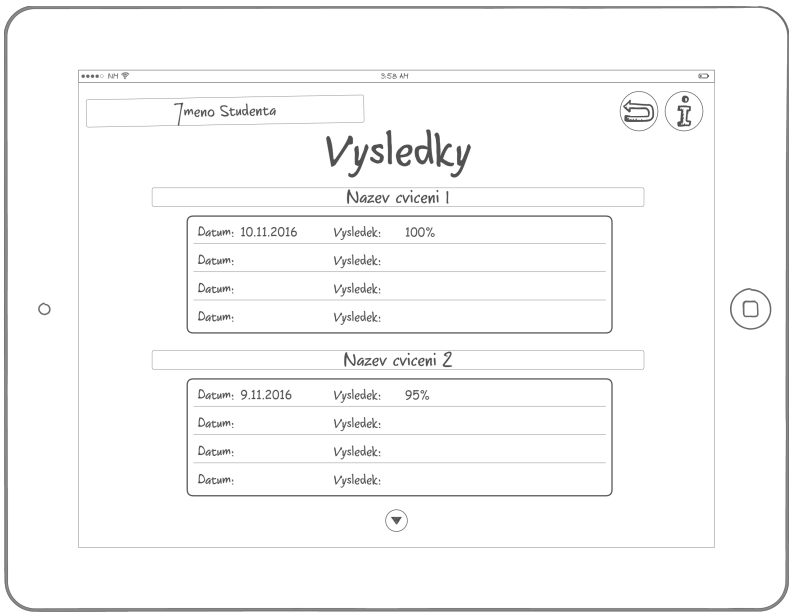
Obrázek 2.7: Zadání

Po jeho přijetí se zobrazí obrazovka 2.8 s potvrzením, že byl výsledek v pořádku přijat a uložen. Podobný dialog se zobrazí, pokud při vyplňování cvičení nastane chyba nebo se student rozhodne ho předčasně ukončit, avšak s varováním, že výsledek uložen nebyl.

Obrázek 2.8: Výsledek modulu

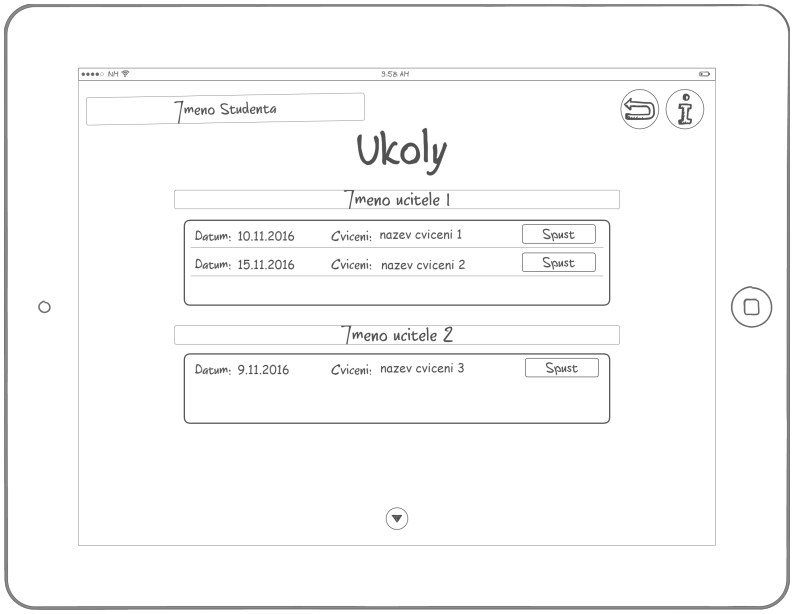
Tím je spouštění cvičení kompletní. Při navigaci z hlavního menu do sekce *Výsledky* se zobrazí obrazovka znázorněna obrázkem 2.9. Zde student vidí všechny výsledky, přehledně rozdělené podle cvičení a seřazené podle data. Ve finální implementaci byly nakonec jednotlivé cvičení umístěny vedle sebe, nikoliv nad sebe. V případě, že by se výsledky jednoho cvičení nevešly na obrazovku tabletu, jsou pak vertikálně posunutelné uvnitř daného boxu.

Obrázek 2.9: Studentovy výsledky



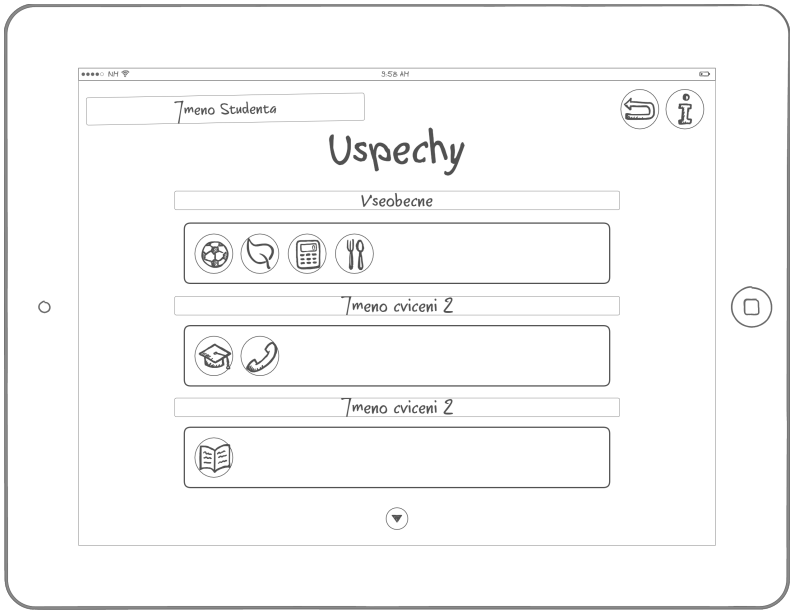
Po navigaci do sekce *Úkoly* se zobrazí obrazovka 2.10, na které jsou podobným způsobem jako u výsledků prezentovány úkoly. Ty jsou setříděné podle zadávajícího učitele a jsou přímo z této obrazovky spustitelné. Po jejich dokončení z ní zmizí a objeví se v sekci s výsledky.

Obrázek 2.10: Zadané úkoly



Jako poslední zde mám připravený přibližný návrh, jak by mohla vypadat obrazovka 2.11 s achievementy i přesto, že se ve finální aplikaci ještě nenalézá. Achievementy, neboli úspěchy jsou zde setříděné podle cvičení pomocí jedinečné malé ikonky.

Obrázek 2.11: Achievementy



Po rozkliknutí ikonky by se zobrazila informace o tom, kdy a za co byl achievement získán, jak je zobrazeno na finálním obrázku 2.12.

Obrázek 2.12: Detail achievementu

Kapitola 3

Testování

Jako každá nově vyvinutá aplikace musel být Dráček před uvažováním o nasazení vhodně otestován. Tento proces probíhal po celou dobu vývoje, především vždy po přidání jakékoliv větší funkcionality, například zobrazování výsledků či offline módu. Ze začátku byl Dráček manuálně testován pomocí Android emulátoru, který je možné spustit na vývojářově počítači a slouží k simulaci Android platformy s přesně zvolenými parametry. V pozdější fázi projektu jsem dostal k dispozici školou zapůjčený tablet Samsung Nexus 10 a veškeré manuální testování probíhalo na něm. Přesně tento typ tabletu bude dostupný také na základní školy Smečno, kam bude Dráček nasazován jako první. K účelu vyzkoušení, zda vše funguje, jak má, byl tedy perfektní.

Doporučených způsobů testování nového softwaru existuje celá řada. V následující části popíši, jakými druhy testů jádro prošlo. Pro tyto účely byl vytvořen následující uživatel:

- **Jméno**: AEinst
- **Heslo**: a

3.1 Jednotkové testy

Jednotkové, či Unit testy jsou určeny k ověření správné funkcionality menšího celku kódu, typicky jedné třídy[31]. Jedná se o plně automatické testy, které je možné spustit naráz stisknutím jediného tlačítka ve vývojářském prostředí. Tento druh je užitečný především pro samotného vývojáře, aby si byl jistý, že například při refaktoringu kódu či rozšiřování stávajících funkcionality nedopatřením nezměnil chování aplikace. Unit testy však nejsou vhodné pro všechny části programu. Obrovské využití mají při kontrole, zda správně funguje výpočetní část kódu, nebo jestli se správně chovají utilitní třídy pro převod formátu dat. Nedají se však použít na testování komunikace mezi komponenty či správnému fungování uživatelského rozhraní.

Zde jsem s jednotkovými testy narazil. Vzhledem k tomu, že jádro je ve své podstatě jeden velký komunikační modul obohacený o již zmíněné uživatelské rozhraní, nebylo zde mnoho věcí, které byly pro jednotkový test vhodné. Vytvořil jsem test na utilitní třídu, která zabezpečuje převod mezi unixovým časovým otiskem a datem používaným v Javě, ale i přes snahu nalézt způsob, jak tento druh testů vhodně využít dále se mi to nepodařilo. Uvažoval jsem použití oblíbené testovací knihovny *Mockito*, která dokáže simulovat chování částí kódu, na kterých testovaná třída závisí, jako jsou různé knihovny apod. Při jejím použití by však byl přínos unit testů téměř nulový, kvůli silnému provázání jádra s použitými knihovnami. Rozhodl jsem se tedy věnovat větší úsilí na testy integrační a uživatelské.

3.2 Integrační testy

Hlavním cílem integračních testů je vyzkoušet, zda spolu jednotlivé samostatné komponenty programu správně komunikují a dohromady zajišťují požadovanou funkcionalitu. Tento druh je pro jádro daleko přínosnější. Testování komunikace však nelze jednoduchou cestou zautomatizovat, tak nezbývalo, než pomocí zapůjčeného tabletu manuálně procházet všechny části uživatelského rozhraní a tím komunikaci vyvolávat a sledovat. Celou dobu testování byl tablet připojen k počítači, kde jsem mohl pomocí nástrojů ve vývojářském prostředí sledovat logované RESTové požadavky i odpovědi a pomocí debug módu také obsah proměnných v jednotlivých důležitých třídách. Kromě těchto věcí jsem samozřejmě sledoval, zda dochází ke korektnímu chování aplikace na tabletu.

Během integračních testů bylo odhaleno několik chyb a nedostatků rozhraní na straně serveru, které se ve spolupráci s Ondřejem Filipem podařilo odstranit. Dále bylo zjištěno, že připravené module Michala Bureše a Mirka Mazla nepočítají s příjmem souborů pomocí třídy FileProvider zmíněné v kapitole o komunikaci jádra a přenosu obrázků. Nutnost použití tohoto konkrétního způsobu zasílání vyšel najevo až při implementaci jádra, tudíž to autorům modulů není možné vyčítat. S pomocí od nového týmu, zabývajícího se tvorbou dalších modulů, se povedlo i tento nedostatek odstranit u modulu *ImageExcludeModule*, který byl dále využit pro testování.

Vyzkoušeny byly všechny průchody aplikací vypsané v části textu zabývajícím se případy užití. Tyto průchody byly vyzkoušeny každý minimálně čtyřikrát a to jak v online, tak offline módu. Internetové spojení bylo přerušováno uměle. Využil jsem k tomu možnost *airplane mode* v tabletu, kterou jsem zapínal v různých částech průchodů, aby byl co možná nejlépe simulován neplánovaný výpadek internetu.

Po dokončení všech oprav proběhlo toto testování 100% úspěšně, všechny průchody proběhly bez chyb a s očekávaným chováním.

3.3 Uživatelské a paměťové testy

Jako poslední druh testů jsem zvolil testy uživatelské a paměťové. Při testování uživatelském jsem manuálně procházel aplikaci a zkoušel, zda všechny akce vyvolávají odpověď, kterou by uživatel očekával a jestli nedochází k výkonnostním problémům. Pokud je výkon aplikace nedostatečný, pocítí to uživatel jako zasekávání a dlouhé čekání na načtení jednotlivých obrazovek. V prvotní implementaci docházelo k automatické synchronizaci všech dat se serverem při každé návštěvě hlavního menu. To se však ukázalo být nepřívětivé, jelikož byl uživatel vždy vyzván k vyčkání na stažení prostředků či upozorněn, že nelze spojení navázat a aplikace běží v offline módu, i když k tomu nemusel být žádný důvod. Na základě těchto testů bylo přidáno manuální aktualizační tlačítko. Dále bylo několikrát lehce upraveno rozložení ovládacích prvků uživatelského rozhraní, například prohlížení výsledků je posouvatelné zleva doprava a výsledky konkrétního cvičení shora dolu, aby nedocházelo nechtěnému posuvu něčeho jiného, než si uživatel přál.

Ještě jsem bohužel neměl možnost jádro otestovat s žáky základní školy, kteří budou aplikaci reálně používat. Toto testování proběhne po vylepšení grafického rozhraní specialistou a doufám, že budu mít možnost se ho účastnit.

Co se týče paměťových testů, sledoval jsem, kolik místa aplikace vyžaduje na samotném tabletu okamžitě po instalaci, po prvním spuštění a synchronizaci se serverem a po čtvrtém průchodu všech případů užití. Výsledky jsou viditelné v tabulce 3.1. Můžeme z nich vyčíst, že po prvotní instalaci se velikost aplikace na zařízení téměř nemění. Ukládané soubory pro offline mód jsou paměťově velice nenáročné a stahované obrázky nejsou třeba ve vysokém rozlišení, nezabírají tedy také mnoho prostoru. S takovým výsledkem jsem byl spokojen.

Tabulka 3.1: Výsledek paměťového testování

	Aplikace [MB]	Data[MB]	Celkem[MB]
Po instalaci	9,290	0,024	9,314
Po prvním spuštění	9,290	0,340	9,630
Po čtvrtém průchodu	9,290	0,340	9,630

Závěr

Hlavním cílem této bakalářské práce bylo vytvoření jádra výukové aplikace pro Android platformu, které bude sloužit jako komunikační a spojovací prvek celé aplikace Dráček II. Bude umožňovat spouštění jednotlivých cvičení, zpracovávat a prezentovat výsledky. Tento cíl se podařilo zcela naplnit. Kromě toho byla implementována podpora běhu bez internetu, která pomůže studentům bez domácího připojení plnit své školní povinnosti bez jakýchkoliv znevýhodnění. Z hlediska celého projektu byl dokončen poslední prvek, který chyběl k tomu, aby mohla být aplikace reálně nasazena. Podařilo se nám vytvořit komplexní výukový systém, který je snadno rozšiřitelný a nabízí vysokou variabilitu ohledně vyučované látky a konkrétních cvičení, které si mohou učitelé sami připravovat. Zároveň byla zachována vysoká úroveň nezávislosti jak na konkrétním cílovém zařízení, kterého je dosaženo klient-server architekturou, tak nezávislosti kódu jednotlivých komponent aplikace.

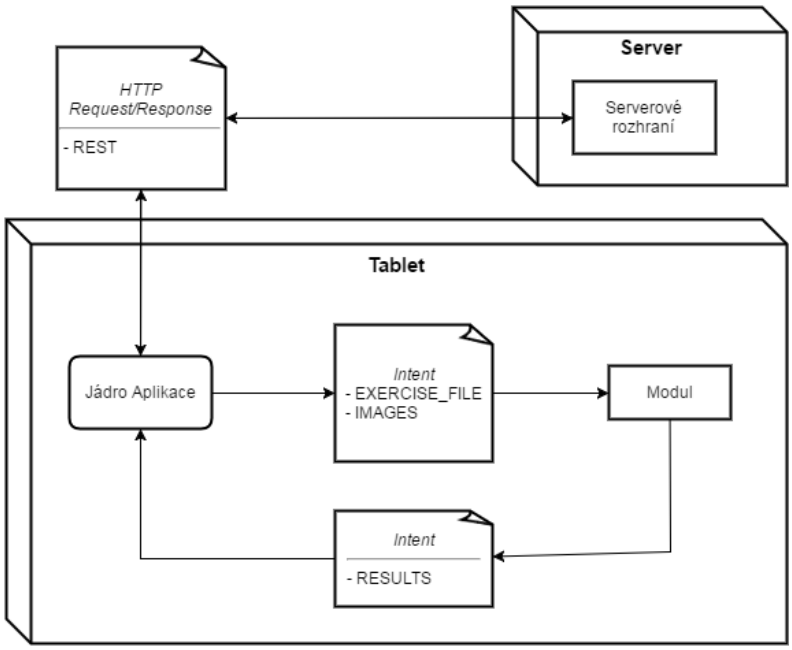
Způsoby řešení

V následující části shrnu, jakým způsobem byly vyřešeny nejpodstatnější části aplikace.

Modularita a komunikace

Jako řešení modularity byl vybrán způsob oddělených APK, který umožnil maximální nezávislost jádra a ostatních komponent Dráčka. Komunikace a výměna výsledků v domluveném formátu se serverem probíhá pomocí RESTu, zatímco předávání informací mezi jádrem a cvičeními umožňují android Intenty. Formát výsledků je jednoznačně určen pomocí XSD, zatímco formát zadání nebylo třeba regulovat. Celkové schéma komunikace je k dispozici na obrázku 3.1.

Obrázek 3.1: Graf komunikace jádra



Offline mód

Pro fungování jádra v offline módu musí být student již do aplikace přihlášen, což je jediný způsob jak zajistit dostačující úroveň bezpečnosti a vyhnout se problémům. Pokud se však ručně neodhlásí, má v offline módu dostupnou veškerou funkcionalitu jako s přístupem na internet ve všech již nainstalovaných modulech. Vyplněné výsledky se se serverem synchronizují po připojení k internetu a spuštění aplikace, či stisknutím speciálního tlačítka.

Osobní rozvoj

Již od začátku se mi tento projekt líbil, protože měl velký potenciál udělat žákům na nejedné škole radost a pomoci jim k lepšímu pochopení látky. Zpříjemňovat výuku nejmladším je dle mého názoru nejlepším způsobem, jak dosáhnout toho, že jim učení nezevšední a že v něm budou rádi pokračovat i ve vyšším věku. Kromě toho mi Dráček dal možnost vyzkoušet si reálnou práci v poměrně rozsáhlém týmu. Od začátku práce na Dráčkovi II až do dnes jsem musel svou práci koordinovat necelou desítkou lidí. Tím byli především O.Filip, M.Mazel a M.Bureš, se kterými jsme od začátku na Dráčkovi pracovali, ale kromě nich samozřejmě také s J.Chludilem jako vedoucím celého projektu, týmem navazujícím na naši práci, zabývajícím se vývojem dalších modulů a

studenty pracující na zpestření a zatraktivnění aplikace.

Díky tomu, že jsem na této práci pracoval až po dokončení jednotlivých modulů i serveru, se pravděpodobně nedalo vyhnout tomu, že bylo potřeba některé již dokončené části upravovat. To se ukázalo být poměrně náročné, vzhledem k časové vytíženosti ostatních členů týmu. Nutnosti některých úprav jsem se dokázal vyhnout, některé jsem s asistencí provedl sám a s některými mi ostatní i přes své vytížení pomohli. Veškeré požadované funkcionality se nakonec podařilo dosáhnout. Komunikace se však ukázala být daleko větším problémem, než jsem původně očekával, což si beru jako ponaučení do budoucna.

Pracovní problémy

Jak jsem detailněji rozebíral v sekci o instalaci modulů, při implementaci jádra jsem narazil na některé omezení a bezpečnostní opatření Android platformy. Jejich kombinace spolu s nedokonalou týmovou komunikací byly hlavním důvodem, proč nebylo prozatím možné implementovat automatické stahování modulů ze serveru. Až bude finálně domluvena licenční a distribuční strategie této aplikace, bude možné tuto část velice jednoduše doimplementovat dle potřeby. Prozatím jádro počítá s manuálním nahráním instalačních balíčků na určené místo, odkud je již samo podle potřeby instaluje.

Dalším problémem se ukázalo být vytvoření korektního implementačního modelu, které bylo jedním z podbodů zadání této práce. Již při začátku práce na jádru v rámci předmětu Softwarový Projekt II jsem si model připravil, avšak během následujícího roku se několikrát modifikovalo zadání, měnily požadavky a vymýšlely nové funkcionality, které by Dráček II mohl nabízet. Zároveň jsem implementaci musel zásadním způsobem přizpůsobit jak cílové platformě Android, tak použitým knihovnám. Atributy všech modelových tříd jsou například přesně dané používanými RESTovými požadavky. Kdybych je nedodržel, nebyla by použitá knihovna schopná automatického převodu Java tříd do JSON formátu. Po všech potřebných úpravách kódu byla realita natolik odlišná od původního modelu, že již neměl žádnou vypovídající hodnotu. Vytvářet nový model až po dokončení implementace by bylo nejen velice obtížné, kvůli provázanosti s knihovnami, ale také ze své podstaty zbytečné. Implementační model má sloužit ke grafickému znázornění vize návrháře aplikace pro programátora. V našem případě jsem však byl v obou těchto rolích já sám. Práci jsem tedy doplnil diagramy znázorňujícími samotný průběh komunikace v rámci celé aplikace a potenciálně problematický kód vysvětlil pomocí komentářů, které by měly k pochopení fungování jádra zcela postačovat.

Budoucnost projektu

Co se týče další budoucnosti projektu, vidím ještě mnoho příležitostí k vylepšení Dráčka. Prvním z nich je již několikrát zmiňované grafické upravení

komponent uživatelského rozhraní, které by mělo proběhnout v co nejbližší době. Dále je jádro připraveno k přidání modulu achivevementů, který bude žáky ještě více motivovat k dosahování co nejlepších výsledku a to zábavným a nenuceným způsobem. Při řešení tohoto problému však bude nutné zohlednit a vyřešit zachování programové nezávislosti jádra a cvičení. Jako poslední možné vylepšení by bylo umožnění přihlášení i jiným způsobem, než je jméno a heslo a přidání mluveného průvodce, což by umožnilo práci s Dráčkem i nejmenším studentům, kteří se ještě nenaučili číst.

I bez těchto úprav se nám však společně podařilo vytvořit povedenou aplikaci a věřím, že bude v blízké době učitelům i žákům zpříjemňovat každodenní výuku.

Literatura

[1] Bradáč, J.: Výuková aplikace Drácek – Jádro klienta. Bakalárská práce, CVUT v Praze, Fakulta informacních technologií, 2013.

[2] Kubišta, P.: Výuková aplikace Drácek – Správa zásuvných modulu. Bakalárská práce, CVUT v Praze, Fakulta informacních technologií, 2013.

[3] Kužela, O.: Výuková aplikace Drácek – Serverová cást. Bakalárská práce, CVUT v Praze, Fakulta informacních technologií, 2013.

[4] Bláha, M.: Výuková aplikace Drácek – Vývoj vybraných modulu. Bakalárská práce, CVUT v Praze, Fakulta informacních technologií, 2013.

[5] Tomšu, R.: Výuková aplikace Drácek – Vývoj vybraných aplikacních zásuvných modulu. Bakalárská práce, CVUT v Praze, Fakulta informacních technologií, 2013.

[6] Filip, O. Výuková aplikace Drácek II – Serverová cást a rozhraní pro ucitele. Bakalárská práce. Praha: Ceské vysoké ucení technické v Praze, Fakulta informacních technologií, 2016.

[7] Mazel, M. Dragon II | Plugins I. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

[8] Bureš, M. Výuková aplikace Drácek II - zásuvné moduly II. Bakalárská práce. Praha: Ceské vysoké ucení technické v Praze, Fakulta informacních technologií, 2016.

[9] Kovarovic, Karel. Gamifikace a personalizace výukové aplikace Drácek. Bakalárská práce. Praha: Ceské vysoké ucení technické v Praze, Fakulta informacních technologií, 2016.

[10] Android Marketplace: Duolingo: Naučte se anglicky. Google Play [online]. [cit. 2016-12-26]. Dostupné z: `https://play.google.com/store/apps/details?id=com.duolingo`

[11] Blog iDnes: Miroslav Hruška, Duolingo - nová internetová stránka na výuku jazyků [online]. iDnes, 2013 [cit. 2016-12-26]. Dostupné z: http://miroslavhruska.blog.idnes.cz/blog.aspx?c=320168

[12] Android Marketplace: Výukové kartičky. Google Play [online]. [cit. 2016-12-26]. Dostupné z: https://play.google.com/store/apps/details?id=com.pmqsoftware.game.childrencards.cz

[13] Android Marketplace: Počítání pro děti. Google Play [online]. [cit. 2016-12-26]. Dostupné z: https://play.google.com/store/apps/details?id=com.kreatur.learningnumbers

[14] Android Marketplace: Jdu do školy!. Google Play [online]. [cit. 2016-12-26]. Dostupné z: https://play.google.com/store/apps/details?id=cz.techsophia.schoolreadiness

[15] Učení v pohodě: Jdu do školy: Výhody aplikace pro děti i rodiče pohledem odborníků [online]. [cit. 2016-12-26]. Dostupné z: http://www.uceni-v-pohode.cz/herne-vyukova-aplikace-pro-predskolni-deti/

[16] Android Developer Documentation: Platform Architecture [online]. [cit. 2017-01-17]. Dostupné z: https://developer.android.com/guide/platform/index.html

[17] OSGI: The Dynamic Module System for Java [online]. [cit. 2017-01-17]. Dostupné z: https://www.osgi.org/

[18] Apache Felix: Documentation [online]. [cit. 2017-01-17]. Dostupné z: http://felix.apache.org/documentation.html

[19] Apache Felix: Apache Felix Framework and Google Android [online]. [cit. 2017-01-17]. Dostupné z: http://felix.apache.org/documentation/subprojects/apache-felix-framework/apache-felix-framework-and-google-android.html

[20] Android Documentation: Intent [online]. [cit. 2017-01-17]. Dostupné z: https://developer.android.com/reference/android/content/Intent.html

[21] Android Documentation: Manifest Permission. [online]. [cit. 2017-01-17]. Dostupné z: http://developer.android.com/reference/android/Manifest.permission.html

[22] Aurora Electronic Data Interchange: EDI resource [online]. [cit. 2017-01-17]. Dostupné z: http://www.auroraedialliance.com/blog/bid/184388/Part-1-The-What-Why-and-How-of-JSON-for-EDI-Integration-Specialists

[23] JSON: Úvod do JSON [online]. [cit. 2017-01-17]. Dostupné z: http://www.json.org/json-cz.html

[24] W3C: Extensible Markup Language (XML) [online]. [cit. 2017-01-17]. Dostupné z: https://www.w3.org/XML/

[25] Android Documentation: FileProvider [online]. [cit. 2017-01-17]. Dostupné z: https://developer.android.com/reference/android/support/v4/content/FileProvider.html

[26] IBM: RESTful Web services: The basics [online]. [cit. 2017-01-22]. Dostupné z: https://www.ibm.com/developerworks/webservices/library/ws-restful/

[27] SitePoint: Networking with Volley in Android [online]. [cit. 2017-01-22]. Dostupné z: https://www.sitepoint.com/volley-a-networking-library-for-android/

[28] Google Developers: Google Play Developer API [online]. [cit. 2017-01-24]. Dostupné z: https://developers.google.com/android-publisher/

[29] Google Code: Android Market Api [online]. [cit. 2017-01-24]. Dostupné z: https://code.google.com/archive/p/android-market-api/

[30] Android Developer Documentation: Manifest Permission [online]. [cit. 2017-01-24]. Dostupné z: https://developer.android.com/reference/android/Manifest.permission.html

[31] ITnetwork: Unit testy v Javě a JUnit [online]. [cit. 2017-01-31]. Dostupné z: http://www.itnetwork.cz/java/pokrocile/java-unit-testy-v-junit

Příloha **A**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `impl`
  - `Core` ..... Git repozitář se zdrojovými kódy aplikace
  - `dragonII_core.apk` ..... Instalační balíček jádra
  - `8.apk` ..... Instalační balíček testovácího modulu
- `text`
  - `latex_source` ..... Zdrojová forma práce ve formátu LaTeX
  - `pavelpat_BP_2017_DragonII.pdf` ..... Text práce ve formátu PDF